UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA AMBIENTAL

TOMAS CARLOTTO

DEVELOPMENT OF A GPGPU ACCELERATED TOOL TO SIMULATE ADVECTION-REACTION-DIFFUSION PHENOMENA IN 2D

Erechim - RS, 2018.

TOMAS CARLOTTO

DEVELOPMENT OF A GPGPU ACCELERATED TOOL TO SIMULATE ADVECTION-REACTION-DIFFUSION PHENOMENA IN 2D

Dissertação apresentada ao Programa de Pós-Graduação em Ciência e Tecnologia Ambiental da Universidade Federal da Fronteira Sul - UFFS como requisito parcial para obtenção do título de Mestre em Ciência e Tecnologia Ambiental, sob a orientação do Prof. Dr. Jose Mario Vicensi Grzybowski e do Prof. Dr. Roberto Valmir da Silva

Erechim - RS, 2018.

UNIVERSIDADE FEDERAL DA FRONTEIRA SUL

ERS 135 - Km 72, 200,
Caixa Postal 764,
CEP 99700-970
Erechim - RS
Brasil

Carlotto, Tomas
   Development of a GPGPU Accelerated Tool to Simulate Advection-Reaction-Diffusion Phenomena in 2D/ Tomas Carlotto. -- 2018.
   145 f.

   Orientador: Jose Mario Vicensi Grzybowski.
   Co-orientador: Roberto Valmir da Silva.
  Dissertação (Mestrado) - Universidade Federal da Fronteira Sul, Programa de Pós-Graduação em Mestrado em Ciência e Tecnologia Ambiental - PPGCTA, Erechim, RS , 2018.

   1. Advection-Reaction-Diffusion Equation. 2. Computational modelling applied to environmental phenomena. 3. Parallel Computing in General Purpose Graphics Processing Unit. I. Grzybowski, Jose Mario Vicensi, orient. II. Silva, Roberto Valmir da, co-orient. III. Universidade Federal da Fronteira Sul. IV. Título.

TOMAS CARLOTTO

DEVELOPMENT OF A GPGPU ACCELERATED TOOL TO SIMULATE ADVECTION-REACTION-DIFFUSION PHENOMENA IN 2D

Dissertação apresentada ao Programa de Pós-Graduação em Ciência e Tecnologia Ambiental da Universidade Federal da Fronteira Sul - UFFS. Para obtenção do título de Mestre em Ciência e Tecnologia Ambiental, defendido em banca examinadora em 09/03/2018.

Orientadores:

Prof. Dr. Jose Mario Vicensi Grzybowski

Prof. Dr. Roberto Valmir da Silva

Aprovado em: 09/03/2018.

Erechim - RS, 2018.

A busca pelo conhecimento é algo gratificante na mesma medida de vossos esforços em fazer o seu melhor trabalho com os recursos que lhe estiverem disponíveis!

Agradecimentos

Agradeço a minha família pela confiança, incentivo e apoio que foram elementos fundamentais em todas as etapas de minha trajetória acadêmica e profissional. Agradeço especialmente aos meus avós (Helio Luiz Carlotto e Isaura Dóris Klin Carlotto), meu pai (Jaques Mozart Klin Carlotto) e tios (Albrecht Volnei Klin Carlotto e Isaias Pablo Klin Carlotto) que sempre confiaram em minha capacidade e me deram forças para alcançar meus objetivos.

Agradeço aos amigos Prof. Jose Mario Vicensi Grzybowski e Prof. Roberto Valmir da Silva pela orientação e fundamental participação no desenvolvimento desta dissertação e pela oportunidade de receber vossos valiosos ensinamentos.

Agradeço aos amigos e colegas pelo afeto e carinho recebido.

Agradeço aos membros da equipe de pesquisa Luiz Augusto Richit, Charline Bonatto e Elvis Prestes pelas contribuições no desenvolvimento deste trabalho.

Enfim, agradeço a todos os professores, colegas e alunos com quem tive a honra de compartilhar experiências, conhecimentos e momentos especiais.

DESENVOLVIMENTO DE UMA FERRAMENTA ACELERADA POR GPGPU PARA SIMULAR FENÔMENOS DE ADVECÇÃO-REAÇÃO-DIFUSÃO EM 2D

Resumo

Os modelos computacionais são ferramentas poderosas para o estudo de sistemas ambientais, desempenhando um papel fundamental em vários campos de pesquisa (ciências hidrológicas, biomatemática, ciências atmosféricas, geociências, entre outros). A maioria desses modelos requer alta capacidade computacional, especialmente quando se considera uma alta resolução espacial e a aplicação em grandes áreas. Neste contexto, o aumento exponencial do poder computacional trazido pelas Unidades de Processamento de Gráficos de Propósito Geral (GPGPU) chamou a atenção de cientistas e engenheiros para o desenvolvimento de implementações paralelas de baixo custo e alto desempenho para modelos ambientais. Neste trabalho, aplicamos computação em GPGPU para o desenvolvimento de um modelo que descreve os processos físicos de advecção, reação e difusão. Esta dissertação é apresentada sob a forma de três artigos. No primeiro, apresentamos uma implementação em GPGPU para a solução da equação de fluxo de águas subterrâneas 2D em aquíferos não confinados para meios heterogêneos e anisotrópicos. Foi implementado um esquema de solução de diferenças finitas com base no método Crank-Nicolson e mostramos que a solução acelerada GPGPU implementada usando CUDA C / C ++ supera a solução serial correspondente implementada em C / C ++. Os resultados mostram que a implementação acelerada por GPGPU é capaz de fornecer aceleração de até 56 vezes no processo da solução usando um computador de escritório comum. No segundo artigo estudamos a aplicação de um modelo de crescimento logístico difusivo (DLG) ao problema de crescimento e regeneração florestal. O estudo foi desenvolvido em duas etapas: (i) Aplicou-se uma metodologia baseada em Comites de Rede Neural Artificial (ANNE) para avaliar a largura da faixa de proteção ripária necessária para filtrar 90% do nitrogênio residual; (ii) O modelo DLG foi calibrado e validado para gerar um prognóstico de regeneração florestal em faixas de proteção ripárias considerando as larguras mínimas indicadas pela ANNE. A solução foi implementada em GPGPU e aplicada para simular o processo de regeneração florestal para um período de quarenta anos na faixa de proteção ripária ao longo do rio Ligeiro, no Brasil. Os resultados da calibração e validação mostraram que o modelo DLG fornece resultados bastante precisos para a modelagem de regeneração florestal. No terceiro artigo, apresenta-se uma implementação em GPGPU para solução da equação advecção-reação-difusão em 2D. A implementação é projetada para ser geral e flexível para permitir a modelagem de uma ampla gama de processos, incluindo características como heterogeneidade e anisotropia do meio. Neste trabalho mostra-se que as simulações realizadas em GPGPU permitem o uso de malhas contendo mais de 20 milhões de pontos (variáveis), correspondendo a uma área de 18.000 km² em resolução de 30m padrão das imagens Landsat.

Palavras - Chave: computação paralela, GPGPU, CUDA, modelagem ambiental.

DEVELOPMENT OF A GPGPU ACCELERATED TOOL TO SIMULATE ADVECTION-REACTION-DIFFUSION PHENOMENA IN 2D

Abstract

Computational models are powerful tools to the study of environmental systems, playing a fundamental role in several fields of research (hydrological sciences, biomathematics, atmospheric sciences, geosciences, among others). Most of these models require high computational capacity, especially when one considers high spatial resolution and the application to large areas. In this context, the exponential increase in computational power brought by General Purpose Graphics Processing Units (GPGPU) has drawn the attention of scientists and engineers to the development of low cost and high performance parallel implementations of environmental models. In this research, we apply GPGPU computing for the development of a model that describes the physical processes of advection, reaction and diffusion. This presentation is held in the form of three self-contained articles. In the first one, we present a GPGPU implementation for the solution of the 2D groundwater flow equation in unconfined aquifers for heterogenous and anisotropic media. We implement a finite difference solution scheme based on the Crank-Nicolson method and show that the GPGPU accelerated solution implemented using CUDA C/C++ (Compute Unified Device Architecture) greatly outperforms the corresponding serial solution implemented in C/C++. The results show that accelerated GPGPU implementation is capable of delivering up to 56 times acceleration in the solution process using an ordinary office computer. In the second article, we study the application of a diffusive-logistic growth (DLG) model to the problem of forest growth and regeneration. The study focuses on vegetation belonging to preservation areas, such as riparian buffer zones. The study was developed in two stages: (i) a methodology based on Artificial Neural Network Ensembles (ANNE) was applied to evaluate the width of riparian buffer required to filter 90% of the residual nitrogen; (ii) the DLG model was calibrated and validated to generate a prognostic of forest regeneration in riparian protection bands considering the minimum widths indicated by the ANNE. The solution was implemented in GPGPU and it was applied to simulate the forest regeneration process for forty years on the riparian protection bands along the Ligeiro river, in Brazil. The results from calibration and validation showed that the DLG model provides fairly accurate results for the modelling of forest regeneration. In the third manuscript, we present a GPGPU implementation of the solution of the advection-reaction-diffusion equation in 2D. The implementation is designed to be general and flexible to allow the modeling of a wide range of processes, including those with heterogeneity and anisotropy. We show that simulations performed in GPGPU allow the use of mesh grids containing more than 20 million points, corresponding to an area of 18,000 km^2 in a standard Landsat image resolution.

Keywords: parallel computing, GPGPU, CUDA, environmental modelling.

Contents

[1]Adapted from [1]

[2]Adapted from [2]

[3]Article in development to be submitted.

List of Figures

List of Tables

List of Symbols

List of symbols in Chapter 2

x - x direction $[L]$

y - y direction $[L]$

z - z direction $[L]$

ρ - Water density

h - Hydraulic head $[L]$

h_s- Topographic elevation $[L]$

h_b- Confining surface elevation $[L]$

h_w- Water height above the topographic $[L]$

K_{xx}- Hydraulic conductivity in the x direction $[\frac{L}{T}]$

K_{yy}- Hydraulic conductivity in the y direction $[\frac{L}{T}]$

K_{zz}- Hydraulic conductivity in the z direction $[\frac{L}{T}]$

K_f - Hydraulic conductivity in forest regions $[\frac{L}{T}]$

K_{nf} - Hydraulic conductivity in regions without forests $[\frac{L}{T}]$

T_{xx}- Transmissivity in the x direction $[\frac{L}{T^2}]$

T_{yy}- Transmissivity in the y direction $[\frac{L}{T^2}]$

$\triangle V$ - Volume variation $[L^3]$

$\triangle x$ - Spatial variation in the x-direction $[L]$

$\triangle y$ - Spatial variation in the y-direction $[L]$

$\triangle t$ - Time variation $[T]$

Q - Source term flow $[\frac{L^3}{T}]$

Q_{calc}- Flow calculated by the model $[\frac{L^3}{T}]$

C_{dr} - Conductance at the boundary between the underground and surface media

S - Storage Coefficient

S_s- Specific storage coefficient $[L^{-1}]$

S_y- Specific flow $[\left(\frac{L^3}{T}\right)\frac{1}{T}]$

g - Acceleration of gravity $[\frac{L}{T^2}]$

α - Compressibility of the geological media $[\frac{L^2}{MLT^{-2}}]$

η - Effective porosity

β - Water Compressibility $[\frac{L^2}{MLT^{-2}}]$

W - Source or extraction term $[L]$

b - Saturated thickness $[L]$

τ - Time variation index

List of symbols in Chapter 3

x - x direction $[L]$

y - y direction $[L]$

t - Time $[T]$

J - Amount of substance per unit area per unit time

D - Diffusion coefficient $[\frac{L^2}{T}]$

u - Concentration or density of forest (Enhanced Vegetation Index (EVI))

u_0- Initial concentration or density of forest (Enhanced Vegetation Index (EVI))

q - Source or extraction term

μ - Porosity of soil

r_u - Logistic growth rate $[L^{-1}]$

k_u - Support capability

ε- Tolerance of error

EVI - Enhanced Vegetation Index

G - Scaling factor

NIR - Reflectance in the near-infrared

RED - Reflectance in red spectral band

$BLUE$ - Reflectance in blue spectral band

C_1 e C_2- Aerosol resistance coefficients

L - Canopy adjustment parameter

MNI_j- mean nitrogen influent for the j - th sub-basin $[ppm]$

N_j^{basis}- Nitrogen load $[ML^{-2}]$

$A_j^{cropland}$- Area of the j - th cropland $[L^2]$

L_j^{river}- Area of the longitudinal section of the river that receives contribution of the j - th sub-basin $[L^2]$

P_j^{runoff} - Runoff $[M^3L^{-2}]$

List of symbols in Chapter 4

x - x direction $[L]$

y - y direction $[L]$

γ - Storage capacity

D_x - Diffusion coefficient in x direction $[L^2T^{-1}]$

D_y - Diffusion coefficient in y direction $[L^2T^{-1}]$

v_x- Velocity in x direction $[LT^{-1}]$

v_y - Velocity in y direction $[LT^{-1}]$

U - Variable that can represent concentration or amount of energy.

$r(U)$ - Term of reaction

W - Source or extraction term

σ - Decay parameter $[T^{-1}]$

r_u - Logistic growth rate $[T^{-1}]$

k - Support capability

τ - Time variation index

μ - Mobility

List of Abbreviations

PDE	Partial differential equation
GPGPU	General Purpose Graphics Processing Units
CUDA	Compute Unified Device Architecture
GPU	Graphic Processing Unit
CPU	Central Processing Unit
BiCGStab	Biconjugate Gradient Stabilized Method
ANA	Water Resources Agency
DEM	Digital Elevation Model
RBS	Riparian Buffer Strips
DLG	Diffusive-Logistic-Growth
DME	Digital elevation model
ANNE	Artificial Neural Network Ensemble
ANN	Artificial Neural Network
VCT	Vegetation cover type
ST	Soil type
MNI	Mean nitrogen inflow
MNE	Mean nitrogen effluent
RE	Removal efficiency
BW	Buffer width
EVI	Enhanced Vegetation Index
RMSE	Root Mean Squared Error
ARD	Advection-Reaction-Diffusion
CRS	Compressed Row Storage
GIS	Geographic Information System
FLOP	Floating-point Operations Per Second
GFLOP	Giga Flop Per Second
GFS	Global Forecast System

Chapter 1

Introduction

The understanding of the dynamic processes that unfold in nature has awakened a certain fascination since the beginning of science. Along with the investigative capacity, human beings developed skills and instruments for observation and measurement and through experimentation they developed ways of isolating and understanding aspects of a real system, generating knowledge that started to support new discoveries and more and more precise forms of describing natural phenomena. All this development has helped to understand that environmental phenomena occurring in nature are complex and that to study them it is necessary to develop abstract ideas that can be represented by models that capture the relationships inherent to environmental phenomena [8].

In the last decades, the need to study environmental systems in an integrated way and with increasing spatial and temporal resolutions has made modeling one of the most powerful tools for scientists and engineers to understand the processes that occur in nature [8, 9, 10, 11, 12]. Along with the mathematical theories and numerical methods, the ability and capacity to perform a large number of calculations in a short period of time is what has allowed the application of the models to the study and understanding of processes. As a result, mathematical modeling and the development of computer technologies have always gone hand in hand. In the last 10 years, the scientific community has seen an unprecedented increase in computing capacity due to the rise of massively parallel computing architectures. To date, even desktop computers when equipped with GPGPUs can perform billions and even trillions of calculations per second and at low cost [13, 14, 15, 16, 17, 18].

These advances have led to the use of parallel computational models in applications ranging from climate change and weather forecasting [11, 19] to hydrological processes [20, 21, 22, 23] and all the way to forest dynamics [24, 2], dispersion of pollutants [19, 25, 18, 26], among many others.

In this master thesis, I applied GPGPU-based parallel computing to explore the solution of a general form of the advection-reaction-diffusion (ARD) equation that can be applied in the modeling of a wide range of environmental problems. The flowchart describing the operation of the model is presented in Fig. 1.1.

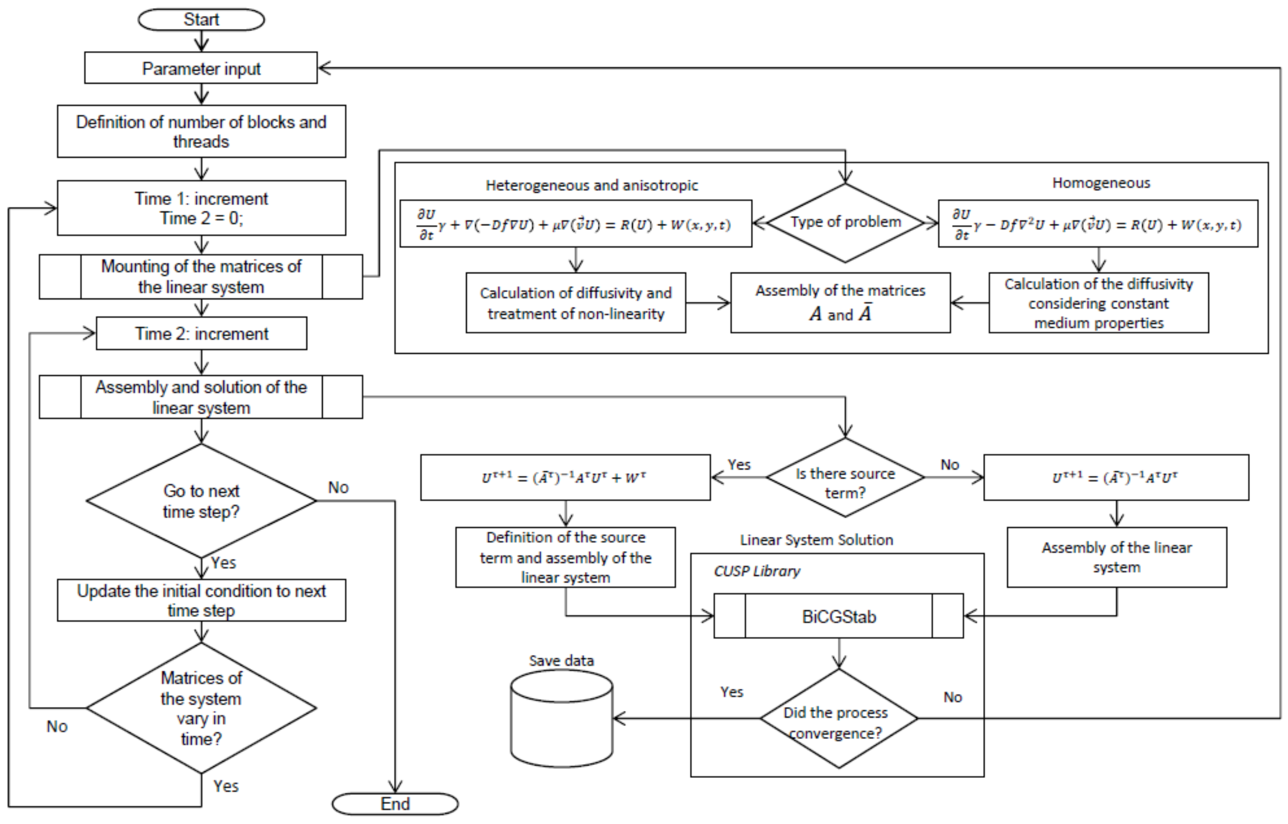

Start
Parameter input
Definition of number of blocks and threads
Time 1: increment
Time 2 = 0;
Mounting of the matrices of the linear system
Time 2: increment
Assembly and solution of the linear system
Go to next time step?
No
Yes
Update the initial condition to next time step
Matrices of the system vary in time?
No
Yes
End
Heterogeneous and anisotropic
Homogeneous
$\frac{\partial U}{\partial t}\gamma + \nabla(-Df\nabla U) + \mu\nabla(\vec{v}U) = R(U) + W(x,y,t)$
Type of problem
$\frac{\partial U}{\partial t}\gamma - Df\nabla^2 U + \mu\nabla(\vec{v}U) = R(U) + W(x,y,t)$
Calculation of diffusivity and treatment of non-linearity
Assembly of the matrices A and $\bar{A}$
Calculation of the diffusivity considering constant medium properties
$U^{\tau+1} = (\bar{A}^{\tau})^{-1}A^{\tau}U^{\tau} + W^{\tau}$
Yes
Is there source term?
No
$U^{\tau+1} = (\bar{A}^{\tau})^{-1}A^{\tau}U^{\tau}$
Definition of the source term and assembly of the linear system
Linear System Solution
CUSP Library
Assembly of the linear system
BiCGStab
Save data
Yes
Did the process convergence?
No

Figure 1.1: Flowchart of the GPGPU-accelerated implementation.

The model presented in Fig. 1.1 consists of a generalized form for the solution of the advection-reaction-diffusion equation by the finite difference method in two dimensions (2D). The spatial and temporal discretization that I developed in this work makes this model adaptable to treat different environmental phenomena, through a computational code that allows to make different combinations involving advection, reaction and diffusion for application in specific cases of environmental systems modeling.

In the flowchart of Fig. 1.1 I show the sequence of operation of the model presenting the main steps for the solution of the ARD equation and the processes and tasks performed in each step. The computational model that I propose allows to treat problems that involve heterogenous and anisotropic means or homogenous means. If the problem involves heterogeneity or anisotropy with variation in the parameters associated with the diffusive term, the problem becomes non-linear and it is necessary to reconstruct the matrices of the linear system at each step of time. If the properties of the medium are constant and homogeneous the matrices are constructed only once and the linear system is solved successively in time.

The assembly of the linear system involves the definition of the term source allowing the use of equations that describe the extraction or insertion of mass or energy in the system to be modeled. The structure of the model allows you to add different types of source or sink terms. A term of logistic growth is already implemented to represent a process of forest regeneration, a term representing the insertion or extraction of groundwater over time and a source term that defines the emission or fixed concentration of a pollutant.

At the end the linear system is solved for each step of time with the biconjugate gradient stabilized method (BiCGStab) made available in the parallel linear algebra library CUSP processed in GPGPU.

The model presented in Fig. 1.1 was tested in three cases: (1) modeling the flow of groundwater in an unconfined aquifer; (2) application of logistic diffusive model (DLG) in forest regeneration modeling and (3) dispersion modeling of pollutants.

The study developed in case 1 was published in the journal Environmental Modeling and Software under the name "GPGPU-accelerated implementation of groundwater flow model in unconfined aquifers for heterogeneous and anisotropic media" [1] and is presented in Chapter 2 of this master thesis. This study was motivated by the relevance and scarcity of numerical approaches to model and solve the flow of groundwater in anisotropic and heterogeneous media using high performance computing. It is known that the evaluation of groundwater and surface water dynamics in real river basins is a challenging issue that requires knowledge of several factors that influence the hydrological cycle, such as the physical and chemical characteristics of the soil and of the environment as a whole. Several studies have been devoted to the development of hydrological models whose objective is to describe the groundwater flow considering different levels of soil saturation and several geological conditions [27, 28, 29].

Among the variety of groundwater models in the literature, one can find models describing the flow of groundwater in river basins [27, 30, 31, 32]. Most commonly, groundwater models are designed to deal with the case of isotropic and homogeneous media [33, 34, 31, 28, 29]. It is known that although assuming the condition of homogeneity and isotropy allows for the simplification of the resulting equations and their solution, this assumption may not be adequate to describe more general situations in which the underground medium is recognized as substantially heterogeneous [35, 8, 33]. In Chapter 2 was generalized the results previously published in the literature that consider the case of homogeneous and isotropic mean, and propose a parallel implementation in GPGPU that allows the solution of the groundwater flow problem to heterogeneous and anisotropic soils.

The case 2 was developed with contributions from other researchers [2, 3], mainly in the part of satellite image processing and treatment to obtain the vegetation index used as input of the DLG model and the development of a sequential implementation of the model which has allowed the verification of the results generated by the parallel model processed in GPGPU, this work was published in the journal Ecological Engineering with the title "Modeling forest regeneration for performance-oriented riparian buffer strips" and composes Chapter 3 of this thesis. The importance of riparian vegetation is recognized in numerous scientific studies which discuss their role in the conservation of species and habitats, the maintenance of water quality and quantity [36, 37, 38, 39, 40, 41, 3, 42], surface and groundwater filtration, erosion mitigation in water bodies, microclimate improvement, sediment retention, nutrient adsorption and dissolved agrochemicals, among others [43, 40]. In this context, the contamination of water by residues of nutrients from agricultural activities is a major concern. This is due to the fact that nutrient concentrations in the environment have a cumulative effect and, therefore, nutrient contamination is one of the main reasons why regeneration and maintenance of riparian strips has become a central argument in favor of water quality.

In Chapter 3 are presented the results obtained by means of the application of a parallel version of the DLG model to the problem of forest recovery. To this end, we consider the problem of forest growth and regeneration in the riparian buffer zone of the Ligeiro river watershed. We show that the GPGPU implementation is capable of providing up to a 50-fold speedup as compared to an equivalent serial implementation.

The case 3 is the application of the ARD model to model the process of dispersion of particulate material smaller than 10 μm (PM_{10}) around a crushing industry, this case is part of a set of tests of the ARD model compiled in an article called "GPGPU-accelerated environmental modeling based on the 2D advection-reaction-diffusion equation" that is being developed for the submitted on Environmental Modeling and Software journal. This study is presented in Chapter 4 in which I describe the development of a model that considers the ARD equation to represent a larger set of environmental problems including contaminant dispersion and decay processes. The development of this model allowed us to expand the discretization schemes used in DLG and groundwater models to obtain a generalized and adaptable implementation that allows the solution of a wide range of environmental problems. The vast applicability of this model is covered in the literature, which include several biological, physical and chemical phenomena that can be modeled with different combinations processes involving reaction, advection and diffusion. Some of the applications of these equations are present in studies of fluid dynamics [23, 44], growth of forests and populations [24], dispersion of pollutants [45, 19, 25, 18, 26], among others [46].

At last, in Chapter 5 I give final remarks to recollect the main contributions of this master thesis and provide some suggestions for future research within these and related topics.

1.1 Objectives

1.1.1 General

- To develop computational tool accelerated by General Purpose Graphical Processing Units (GPGPU) for the simulation of environmental processes involving advection, reaction and diffusion.

1.1.2 Specific

- To develop and to implement a numerical solution scheme for the equation of groundwater flow in heterogeneous and anisotropic media in two dimensions.
- To implement numerical solution schemes for the characteristic equation of the two-dimensional diffusive logistic growth process.
- To obtain a discrete form of the 2D Advection-Reaction-Diffusion equation with a generalized scheme that allows the treatment of a wide variety of physical problems using an unified computational approach implemented in GPGPU with NVIDIA ® CUDA technology.
- To test the tool developed in the simulation of problems of groundwater flow, diffusive logistic growth and dispersion of pollutants.

Chapter 2

A GPGPU-accelerated implementation of groundwater flow model in unconfined aquifers for heterogeneous and anisotropic media[1]

2.1 Introduction

Spatially distributed models are widely recognized as important tools for the understanding of groundwater dynamics [34, 47, 48, 49, 17]. They are widely applicable in environmental analysis of flooded areas, pollutant contamination of aquifers, wells and reservoirs [50, 31, 51, 52, 49, 53, 28, 54, 55, 34, 56, 29, 17]. The implementation of computer models has benefited from the rapid growth in computational capabilities observed over the last thirty years or so [28, 34]. More recently, the rapid emergence of parallel computing platforms based on GPGPU (General Purpose Graphics Processing Units) has provided an entirely new perspective regarding the processing capabilities of personal computers, thus attracting the attention of the modelling community [57, 58, 21, 59, 60, 17]. In this context, hydrological models are particularly suitable for massively parallel frameworks due to the state-of-the-art of solution techniques for sparse linear systems via domain decomposition methods [61, 62].

The assessment of groundwater and surface water dynamics in real basins is a challenging issue which requires knowledge of several factors that influence the hydrological cycle, such as the physical and chemical characteristics of the soil and of the environment as a whole. A number of studies have been devoted to the development of hydrologic models whose aim is to describe groundwater flow while considering different levels of soil saturation and diverse geologic conditions [27, 28, 29]. Among the variety of groundwater models in the literature, one can find models that describe groundwater flow in basins [32, 31, 27, 30]. Most commonly, groundwater models are designed to deal with the case of isotropic and homogeneous media [33, 34, 28, 29, 31] . Although the assumptions on homogeneity and isotropy permit the simplification of the resulting equations and their solution, they might not be adequate in describing

[1]Adapted from [1]

more general situations in which the flow media is recognized to be substantially heterogeneous [35, 8, 33].

Regarding the application of high-performance computing to simulations of water and environment-related phenomena, several publications have shown the advantages of parallel computing frameworks [21, 60, 59, 49, 63, 47, 64, 48, 17]. The increased efficiency and reduced computational time of parallel frameworks allow for the simulation of physically-based models over large areas with finer grid resolutions [63, 21, 60, 17]. In addition, the reduced computing time enables rapid modeling and analysis during emergency situations in the wake of environmental disasters [47]. In this regard, high-performance computer models can play a fundamental role in reducing the response time following environmental disasters, thus broadening the perspectives of achieving effective engineering solutions for emerging problems.

Recent publications have shown that parallelized computer models are a feasible and affordable means of rapidly assessing engineering solutions. As time is a major concern in the calibration of hydrologic models, the parallelization of calibration algorithms in CPUs has provided increased speedup and scalability [47]. In this context, GPU implementations are expected to provide even greater speedups [47] due to the increasing number of processors and the amount of memory available. Indeed, in Ref. [21], CUDA$^{\text{TM}}$-based solvers were applied to transient groundwater flow problems and achieved roughly a 4-fold speedup. Another noticeable fact of parallel implementations is the finer-grained resolutions for computational mesh grids. Take for instance the parallel implementation of GCSFlow (GPU-based Conjunctive Surface Subsurface Flow Model, which permitted the computation of surface and subsurface water flow for a domain with a topographic resolution of $1.2m \times 1.2m$ [17]. Another example of parallelization related to hydrologic models was the implementation of MODFLOW [65, 33], in which GPU methods were shown to outperform multi-CPU methods [60]. In this specific case, the parallelized version of MODFLOW was found to provide a 10-fold speedup in relation to the serial one [60]. This indicates that the application of GPGPU is a promising path to enhancing massive processing of scientific data.

In this paper, we develop a GPGPU-accelerated implementation of the groundwater flow in unconfined aquifers for anisotropic and heterogeneous media. We generalize results previously published in the literature which consider the case of homogeneous and isotropic media and propose a massively parallel implementation that enables the solution of the groundwater flow problem for heterogeneous and anisotropic soils. There are two main difficulties associated with the numerical solution of this model: (i) the resulting equation is nonlinear, as the hydraulic conductivity depends on the hydraulic head, and (ii) due to the fact that the parameters are state-dependent, the matrix of the corresponding linear system has to be redefined at each time step, which is time-consuming. These issues were solved by (i) considering a quasi-linear approach in which the hydraulic conductivity variable parameter is one step behind the hydraulic conductivity, and (ii) developing a dedicated kernel to redefine the matrix at each time step. The model is discretized using the Crank-Nicolson finite-difference scheme [66, 67] and then solved using a CUDA^{TM} C/C++ implementation based on CUSP library [68]. The main objective of this research is to provide an efficient and sufficiently general parallel implementation of the groundwater flow model. As a means of testing the solution speedup, we consider the groundwater flow problem for different grid resolutions running CUDA^{TM} C/C++ parallel code and then its serial counterpart in C/C++. The results show that the CUDA^{TM} C/C++ parallel implementation achieves up to a 56-fold speedup in relation to the serial one. The outline of the paper is as follows: the materials and methods are presented in section 2.2; the GPGPU implementation performance results and application to a real basin are found in section 2.3; the results are discussed in section 2.4 and put into perspective with similar studies from the literature; final remarks and future perspectives are dealt with in section 2.5. future perspectives.

2.2 Materials and methods

In this section, we present the model equations, the discretization process and the solution method for the resulting linear system using CUDA^{TM}.

2.2.1 The equations for groundwater flow in unconfined aquifers

The groundwater flow can be modelled as the transient behavior of the hydraulic heads. With this in mind, Ref. [69, 70, 71] were studied in detail on the basis of the continuity equation and Darcy's law. Considering the hydraulic conductivities K_{xx}, K_{yy}, K_{zz}, Darcy's equation and the continuity equation for mass, a 3D model for groundwater flow for the hydraulic head h can be written as

$$\left(\frac{\partial}{\partial x}\left(\rho.K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(\rho.K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(\rho.K_{zz}\frac{\partial h}{\partial z}\right)\right).\triangle V+\rho.Q.\triangle V=S_s.\frac{\partial h}{\partial t}\triangle V \quad (2.1)$$

in which ρ is the fluid density, $\triangle V$ is the variation in volume, Q is associated with a sink/source term and S_s is the storage term, given by

$$S_s=\rho.g.[\alpha+\eta.\beta]$$

where g is gravity; α is the compressibility of the geologic media, η is the soil porosity and β is the water compressibility [72]. In order to make the model tractable, the following assumptions can be adopted.

- *The fluid is incompressible*, which means the fluid density is constant. This is a reasonable assumption since the model is not dealing with coastal zones (where there is a mix of fresh water and salt water within the soil layers) nor groundwater contamination (where certain contaminants are denser than the groundwater).
- *The soil is saturated.* This assumption is justified by the fact the model is represents the hydraulic head surface dynamics, which takes place in saturated soil.
- *The flow is laminar within the range of validity of Darcy's equation.* This assumption represents most of the groundwater flow. Although turbulent flow is observed in certain cases, such as, pumping in wells and very coarse-grained soils, it is not meant to be represented by the proposed model.

Under these conditions, Eq. 2.1 can be rewritten as

$$\frac{\partial}{\partial x}\left(K_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}\frac{\partial h}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_{zz}\frac{\partial h}{\partial z}\right)+W(x,y,z,t)=S_s.\frac{\partial h}{\partial t} \tag{2.2}$$

where $W(x,y,z,t)$ is a source term dependent on time and position. From Eq. 2.2, one can derive a 2D equation of groundwater flow by considering the integral over the vertical dimension z, given as

$$\bar{h}=\frac{\int_0^b hdz}{\int_0^b dz}=\frac{\int_0^b hdz}{b} \tag{2.3}$$

which gives

$$\frac{\partial}{\partial x}\left(K_{xx}b\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_{yy}b\frac{\partial h}{\partial y}\right)+\left(K_{zz}\frac{\partial h}{\partial z}\right)|_{z=b}-\left(K_{zz}\frac{\partial h}{\partial z}\right)|_{z=0}+\cdots$$

$$+W(x,y,t)=S_s b\frac{\partial \bar{h}}{\partial t} \tag{2.4}$$

In confined aquifers, b can be considered as a constant for each point of the grid. However, for unconfined aquifers, the height of the saturated layer b varies as a function of the hydraulic head. Thus, the resulting equation for groundwater flow in anisotropic and heterogeneous media, in its 2D formulation, can be written as

$$\frac{\partial}{\partial x}\left(T_{xx}\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(T_{yy}\frac{\partial h}{\partial y}\right)+W(x,y,t)=S\frac{\partial h}{\partial t} \tag{2.5}$$

where the parameters $T_{xx}=K_{xx}b$, $T_{yy}=K_{yy}b$ and $S=S_sb$, in which T_{xx}, T_{yy} are transmissivities in the directions x, and y, respectively, S_s is the specific storage and b is the height of the saturated layer, while $W(x,y,t)$ stands for the source/sink term. The source/sink term W in the model describes the inflow or outflow of water in the control volume. In our approach, W is the outflow of groundwater to the surface, which will start/maintain the surface flow. In this case, W is a sink for the groundwater model and a source for the surface water model, thus contributing to surface flow rate. To model this process and for it to remain physically meaningful in the absence of the surface water model, the following assumptions are made: (i) groundwater outflow to the surface will become overland flow in a given cell if the hydraulic head in this cell is larger than the terrain elevation, and (ii) the groundwater outflow equals to the volume of water computed by the model multiplied by the specific flow rate of the underground media, S_y. This multiplication guarantees that the water outflow volume represents the volume that can be withdrawn from the underground media taking into account the capillary force. Using these considerations, one can define the water outflow from the control volume as

$$\begin{cases} \text{If } h_{s_{i,j}} \geq h_{i,j}^{\tau} & \text{then } W_{i,j}^{\tau} = 0 \\ \text{If } h_{s_{i,j}} < h_{i,j}^{\tau} & \text{then } W_{i,j}^{\tau} = (h_{s_{i,j}} - h_{i,j}^{\tau}) \end{cases} \tag{2.6}$$

where $W_{i,j}^{\tau} < 0$ indicates the amount of water flowing out of the underground media and $W_{i,j}^{\tau} = 0$ indicates that there is no outflow. The height of the water column, which will compose the baseflow, is given as

$$h_{w_{i,j}}^{\tau} = -W_{i,j}^{\tau} S_y \tag{2.7}$$

The base flow is then given by

$$Q_{calc} = C_{dr} \cdot \frac{(h_{w_{i,j}}^{\tau} \Delta x \Delta y)}{\Delta t} \tag{2.8}$$

where C_{dr} is the draining capacity of the boundary between the underground media and the surface, Δx and Δy are the grid step sizes in the x, y-direction, respectively and Δt is the time step. As this formulation does not consider the surface flow dynamics, it means that we are making the implicit assumption that the groundwater outflow will have enough time to reach the basin outlet.

2.2.2 Discretization of the model

The major challenge in the modelling of unconfined aquifers is associated with the description of the moving physical boundary. Whereas the upper and lower boundaries do not change for confined aquifers, which results in a fixed saturated layer height for each point of the mesh, only the bottom boundary remains unchanged for unconfined aquifers. The upper boundary is a function of the hydraulic head and, as such, varies over time and space. Besides, the transmissivities are a function of the hydraulic head, meaning that the problem becomes nonlinear and consequently affects the solution of the resulting equations. In this case, approximate solutions can be obtained by assuming that the height of the saturated layer in time $\tau + 1$ is a function of hydraulic head in time τ, that is $b^{\tau+1} = b(h^{\tau})$. Thereby, the expression of the derivatives can be written as $\frac{\partial}{\partial x}\left(K_{xx}b(h^{\tau})\frac{\partial h^{\tau+1}}{\partial x}\right)$ under the assumption that the transmissivities do not change considerably from one time step to the next, which is fine if the time steps are small, i.e., there is very little variation in the hydraulic head from one time step to the other. In this context, the size of the time step gains importance since such variations have to be checked for consistency with the assumption.

From this formulation, one obtains the linearized Boussinesq equation [73, 74], given by

$$\frac{\partial}{\partial x}\left(K_{xx}b(h)\frac{\partial h}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_{yy}b(h)\frac{\partial h}{\partial y}\right) + W(x,y,t) = S_y\frac{\partial h}{\partial t} \tag{2.9}$$

where K_{xx}, K_{yy} are hydraulic conductivities in the x, y-direction, respectively, h is the hydraulic head, $b(h)$ is the height of the saturated layer as a function of hydraulic head, W is the source/sink term and S_y is the specific flow rate, more commonly known as effective porosity. Figure 2.1 illustrates the relationship between topographic elevation (h_s), the confining surface (h_b) and the hydraulic head (h).

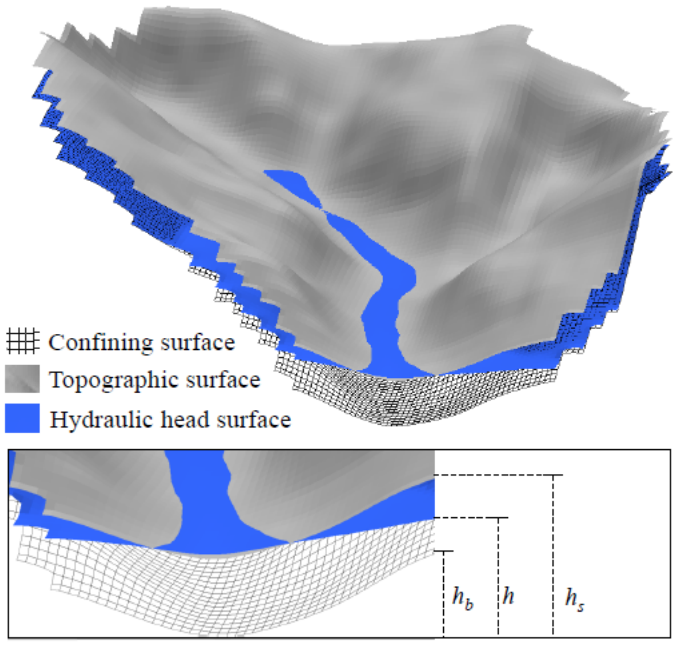


Figure 2.1: Illustration of the relation among topographic elevation (h_s), confining surface elevation (h_b) and hydraulic head (h).

On the basis of Fig.2.1, one can define the height of the saturated layer in time τ as [33, 75]:

$$b^{\tau} = (h^{\tau} - h_b) \tag{2.10}$$

The height of the saturated layer, b, is computed so that it satisfies Eqs. 2.11, 2.12, 2.13, which limit the saturated layer between the topographic elevation and the elevation of the confining surface [33, 75]. Following this reasoning, one obtains the following conditions:

$$\text{If } h_s > h^\tau > h_b \text{ then } b^\tau = (h^\tau - h_b) \tag{2.11}$$

$$\text{If } h_s < h^\tau \text{ then } b^\tau = (h_s - h_b) \tag{2.12}$$

$$\text{If } h^\tau \leq h_b \text{ then } b^\tau \simeq 0 \tag{2.13}$$

As previously mentioned, in the case of groundwater flow in unconfined aquifers, the physical boundary moves with time and space, since it is a function of the hydraulic head. In this case, the height of the saturated layer must be calculated by means of Eq. 2.9 for each point in the mesh in each time step. Applying the approximation of derivatives by parts, one obtains

$$\begin{aligned}
&\frac{1}{\triangle x_j}\left[T^\tau_{xx_{i,j+\frac{1}{2}}}\left(\frac{h^{\tau+1}_{i,j+1} - h^{\tau+1}_{i,j}}{\triangle x_{j+\frac{1}{2}}}\right) - T^\tau_{xx_{i,j-\frac{1}{2}}}\left(\frac{h^{\tau+1}_{i,j} - h^{\tau+1}_{i,j-1}}{\triangle x_{j-\frac{1}{2}}}\right)\right] + \cdots \\
&+\frac{1}{\triangle y_i}\left[T^\tau_{yy_{i+\frac{1}{2},j}}\left(\frac{h^{\tau+1}_{i+1,j} - h^{\tau+1}_{i,j}}{\triangle y_{i+\frac{1}{2}}}\right) - T^\tau_{yy_{i-\frac{1}{2},j}}\left(\frac{h^{\tau+1}_{i,j} - h^{\tau+1}_{i-1,j}}{\triangle y_{i-\frac{1}{2}}}\right)\right] + \cdots \\
&+ W^{\tau+1}_{i,j} = \frac{S^\tau_{y_{i,j}}}{\triangle t}(h^{\tau+1}_{i,j} - h^\tau_{i,j})
\end{aligned} \tag{2.14}$$

where the transmissivities are defined as

$$T^\tau_{yy_{i+\frac{1}{2},j}} = K_{yy_{i+\frac{1}{2},j}} b^\tau_{i+\frac{1}{2},j} \tag{2.15}$$

$$T^\tau_{yy_{i-\frac{1}{2},j}} = K_{yy_{i-\frac{1}{2},j}} b^\tau_{i-\frac{1}{2},j} \tag{2.16}$$

$$T^\tau_{xx_{i,j+\frac{1}{2}}} = K_{xx_{i,j+\frac{1}{2}}} b^\tau_{i,j+\frac{1}{2}} \tag{2.17}$$

$$T^\tau_{xx_{i,j-\frac{1}{2}}} = K_{xx_{i,j-\frac{1}{2}}} b^\tau_{i,j-\frac{1}{2}} \tag{2.18}$$

Following the same procedure as before, Eq. 2.14 can be rewritten in the simplified form

$$\begin{aligned}
&F^\tau_{i,j}(h^{\tau+1}_{i,j+1} - h^{\tau+1}_{i,j}) - D^\tau_{i,j}(h^{\tau+1}_{i,j} - h^{\tau+1}_{i,j-1}) + M^\tau_{i,j}(h^{\tau+1}_{i+1,j} - h^{\tau+1}_{i,j}) + \cdots \\
&- B^\tau_{i,j}(h^{\tau+1}_{i,j} - h^{\tau+1}_{i-1,j}) + W^t_{i,j} = \frac{S_{y_{i,j}}}{\triangle t}(h^{\tau+1}_{i,j} - h^\tau_{i,j})
\end{aligned} \tag{2.19}$$

where $W_{i,j}^{\tau}$ is given by

$$W_{i,j}^{\tau} = S_y(h_{i,j}^{\tau} - h_{s_{i,j}}) \tag{2.20}$$

and the further parameters are defined as follows:

$$B_{i,j}^{\tau} = \frac{\left[\frac{2T_{yy_{i,j}}^{\tau} T_{yy_{i-1,j}}^{\tau}}{T_{yy_{i,j}}^{\tau} \triangle y_{i-1} + T_{yy_{i-1,j}}^{\tau} \triangle y_i}\right]}{\triangle y_i} \tag{2.21}$$

$$D_{i,j}^{\tau} = \frac{\left[\frac{2T_{xx_{i,j}}^{\tau} T_{xx_{i,j-1}}^{\tau}}{T_{xx_{i,j}}^{\tau} \triangle x_{j-1} + T_{xx_{i,j-1}}^{\tau} \triangle x_j}\right]}{\triangle x_j} \tag{2.22}$$

$$F_{i,j}^{\tau} = \frac{\left[\frac{2T_{xx_{i,j}}^{\tau} T_{xx_{i,j+1}}^{\tau}}{T_{xx_{i,j}}^{\tau} \triangle x_{j+1} + T_{xx_{i,j+1}}^{\tau} \triangle x_j}\right]}{\triangle x_j} \tag{2.23}$$

$$M_{i,j}^{\tau} = \frac{\left[\frac{2T_{yy_{i,j}}^{\tau} T_{yy_{i+1,j}}^{\tau}}{T_{yy_{i,j}}^{\tau} \triangle y_{i+1} + T_{yy_{i+1,j}}^{\tau} \triangle y_i}\right]}{\triangle y_i} \tag{2.24}$$

By grouping like terms from Eq. 2.19 and adapting the resulting equation to the Crank-Nicolson scheme [66, 67], the following implicit-solution equation is obtained:

$$\begin{gathered}
\left[1 + \lambda_{i,j}\left(F_{i,j}^{\tau} + D_{i,j}^{\tau} + M_{i,j}^{\tau} + B_{i,j}^{\tau}\right)\right] h_{i,j}^{\tau+1} + \cdots \\
-\lambda_{i,j}\left[F_{i,j}^{\tau} h_{i,j+1}^{\tau+1} + D_{i,j}^{\tau} h_{i,j-1}^{\tau+1} + M_{i,j}^{\tau} h_{i+1,j}^{\tau+1} + B_{i,j}^{\tau} h_{i-1,j}^{\tau+1} - W_{i,j}^{\tau}\right] \\
= \\
\left[1 - \lambda_{i,j}\left(F_{i,j}^{\tau} + D_{i,j}^{\tau} + M_{i,j}^{\tau} + B_{i,j}^{\tau}\right)\right] h_{i,j}^{\tau} + \cdots \\
+\lambda_{i,j}\left[F_{i,j}^{\tau} h_{i,j+1}^{\tau} + D_{i,j}^{\tau} h_{i,j-1}^{\tau} + M_{i,j}^{\tau} h_{i+1,j}^{\tau} + B_{i,j}^{\tau} h_{i-1,j}^{\tau} - W_{i,j}^{\tau}\right]
\end{gathered} \tag{2.25}$$

where

$$\lambda_{i,j} = \frac{\triangle t}{2S_{y_{i,j}}} \tag{2.26}$$

It should be rememberedl that the matrix structures can be rewritten as

$$A^{\tau}h^{\tau+1} = \delta^{\tau}h^{\tau} + W^{\tau} \tag{2.27}$$

and an explicit expression for the hydraulic heads in time $\tau + 1$ is given by Eq. 2.28, which reads

$$h^{\tau+1} = (A^{\tau})^{-1}(\delta^{\tau}h^{\tau} + W^{\tau}) \tag{2.28}$$

Remember also that the conditions under which the problem is formulated guarantee that the linear system in Eq. 2.27 has a unique solution. Finally, Eq. 2.28 represents the implicit form of the solution of Eq. 2.9. The above formulation can be applied in the study of groundwater flow in unconfined aquifers in heterogeneous and anisotropic media. The solution requires that the matrix A^{τ} is mounted at every time step, since it is dependent on the system's unknowns.

2.2.3 Solution of the system: CUDA$^{\text{TM}}$ C/C++ implementation

The code for the formulation and solution of the equations was developed in Visual Studio Community 2013 [76] with CUDA$^{\text{TM}}$ Toolkit 8.0 [77] and the CUSP library [68]. The details of the implementation are given in the following subsections.

Generation and updating of time-dependent system matrix and vectors

Recall from the discretization process that the terms $A^{\tau}, \delta^{\tau}, W^{\tau}$ from the linear system in Eq. 2.27 must be redefined at each time step due to the dependence of transmissivities on the hydraulic head, h^{τ}. Thus, before the linear system in Eq. 2.27 is solved within a time step, its terms must be updated. In other words, the matrix A^{τ} and the vectors δ^{τ}, W^{τ} must be mounted at each time step. Since this is massively repetitive task, a CUDA$^{\text{TM}}$ kernel was developed to deal with the calculation of each nonzero entry of the matrix and vectors. The kernel is made available for download along with the solution code [78].

Solving the linear system

The linear system was solved iteratively at each time step by means of the Biconjugate gradient stabilized method (BiCGStab) [79]. The advantages that BiCGStab offers as an iterative solver are the reduced computational effort and memory requirements [79]. We applied the implementation of BiCGStab available in the CUSP library for CUDA$^{\text{TM}}$ [68].

Description of the hardware equipment

The solution was performed using an Intel®$i7^{\text{TM}} - 7700K$ CPU with $16GB$ RAM equipped with a NVIDIA®GeForce®GTX 1060 GPU card.

2.3 Results

The performance comparison between pipeline and parallel implementations of the groundwater flow model are presented in this section. In addition, as a demonstrative example, an application of the model in the study of the behavior of the variation of soil saturation during an extended drought is also presented.

2.3.1 Performance of the GPGPU-accelerated implementation

The details of the performance of the GPGPU-accelerated implementation are presented in Figure 2.2 and Table 2.1. The results show that the GPGPU-accelerated solution yielded up to a 56-fold speedup in relation to the solution provided by a conventional serial code. The considerable speedup can be attributed to the parallelization of the mounting of the matrices and vectors of the linear system at each time step. It was observed that this process is quite time-consuming in the serial version of the code. Remember that the hydraulic transmissivities are dependent on the hydraulic head, so the system matrix and vectors, $A^{\tau}, \delta^{\tau}, W^{\tau}$, must be redefined at each time step τ.

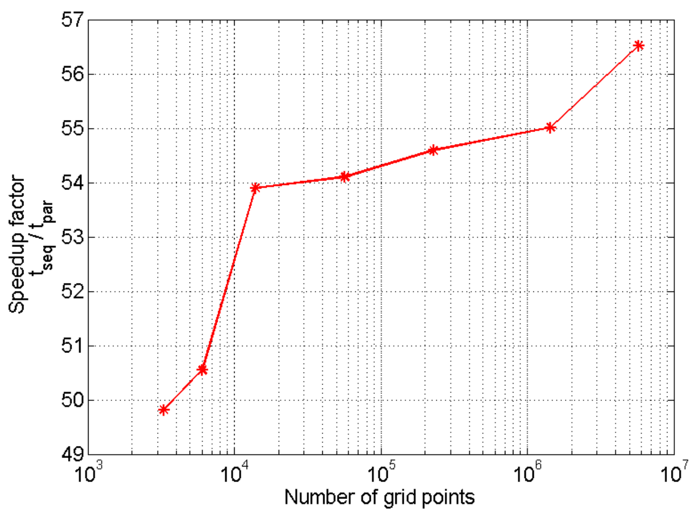


Figure 2.2: Speedup factor of the parallel CUDA$^{\text{TM}}$ C/C++ implementation of the groundwater model in relation to the serial C/C++ implementation.

2.3.2 Application: evaluation of the behavior of the basin during a drought

The relationship between the variations on the level of the unconfined aquifer and the fluctuations in the water levels in water bodies is an important matter in Hydrology. It has important consequences upon the analysis of the potential of storage and transport in aquifers [74]. Such

Table 2.1: Results from pipeline and parallel solutions of the groundwater flow model.

Grid points	Resolution (m)	Time (s) - serial	Time (s) - parallel	Speedup
3.30×10^3	1,200	7.7	0.16	49.8x
6.05×10^3	900	14.2	0.28	50.5x
1.39×10^4	600	32.9	0.61	53.9x
5.68×10^4	300	132.6	2.45	54.1x
2.29×10^5	150	533.3	9.77	54.6x
1.44×10^6	60	3,379.0	61.42	55.0x
5.79×10^6	30	13,651.0	241.53	56.5x

analyses can support the planning of water supply systems and guide the installation of irrigation systems. Furthermore, they allow water supply managers to estimate the optimal time to adopt measures and strategies to control or restrict the use of water.

The idea behind the simulation of a drought is to evaluate the level of the unconfined aquifer during the time when there are no inputs (rain). This is one very common application of groundwater flow models as applied to the management of catchment basins. By means of this analysis, the quantity of water and the time horizon in which the required output flow will occur can be estimated, thus enhancing the technical basis for decision-making.

In this section, the GPGPU-accelerated implementation of the groundwater flow model for unconfined aquifers is calibrated, validated and applied to the study of the Apuaê basin, in Brazil. The objective is to show that such a study can be straightforwardly conducted by means of the CUDA$^{\text{TM}}$ C/C++ implementation presented in this paper. The hardware requirement is an ordinary office computer equipped with an NVIDIA®GPU card and the CUDA$^{\text{TM}}$ Toolkit 8.0 or later versions [77], as described in Section 2.2.

Characterization of the basin

The study basin is delineated assuming its outlet at the river level gauge Passo Colombelli ($27^o33'43''S$, $51^o51'28''W$), which is maintained by the Water Resources Agency (ANA). This outlet location defines a contribution area of 3.660 km^2. Since the main river in the basin is called the Apuaê River, we have defined herein the basin name to be the Apuaê Basin. The Apuaê Basin is part of the Uruguay Basin, an important basin for the generation of hydroelectric power, and which is located in the countries of Brazil, Argentina and Uruguay. In Brazil, the Uruguay Basin is located in the states of Santa Catarina and Rio Grande do Sul. Figure 2.3 depicts the location of the Apuaê Basin in the Brazilian part of the Uruguay Basin, as well, its elevation. Elevations in the basin range from 387 m to 929 m above sea level. Slopes are distributed according to the following percentages of the basin area: 67.72% $(0^o - 22^o)$, 26.34% $(22^o - 44^o)$ and 5.94% $(45^o - 66^o)$. The valleys are V-shaped with hillslopes heights ranging from 100 m to 300 m [80]. The spatial distribution of soil depths in the basin were estimated by means of the well depths database (72 wells) acquired from the Groundwater System Information [81] and the methodology presented in Ref. [82]. Thus, soil depths in the basin range from 8 to 230 meters. There are three classes of soil in the basin: 1) Cambisol soils (middle-northern to northeastern part), which are characterized by an inexpressive B horizon differentiation in the soil profile and are mainly agricultural areas; 2) Latosol soils, which cover the majority of the basin (northwestern and middle-southern parts). This class of soil has no distinct horizons and its color varies from red to yellowish-red. These soils constitute the largest class of territorial expression and agricultural potential of the country, being used for various crops, reforestation and pasture [83]; and 3) Nitosol class (southern basin border) present clayey to very clayey soils with slight increase of clay in the subsurface horizon [84]. The Apuaê River is 208 km in length with its riverhead and outlet at elevations of 812 m and 387 m, respectively. The average discharge recorded (1939 - 2014) at the basin outlet is $99.65 m^3 s^{-1}$. The average annual precipitation throughout the basin is around 1,760 mm (1954 - 2014). The rainy season takes place during the months of September, October and November and the dry season during the months of March, April and May. The runnoff coefficient is 0.48, i.e., about 52% of the total precipitation throughout the basin is lost to the atmosphere due to the evapotranspiration process. This ratio is consistent with data from non-urbanized basins. The average, maximum and minimum temperatures throughout the basin are $17.5^o C$, $23.6^o C$ and $13.2^o C$, respectively. The Apuaê Basin is located within a region of intense agricultural activity based the cultivation of soybean and corn during the summer months, and wheat and oat crops during the winter months. Only 36% of the total area is covered by forests (see Figure 2.3). This fact raises concern for water resources management as the water demand for irrigation and water supply increases.

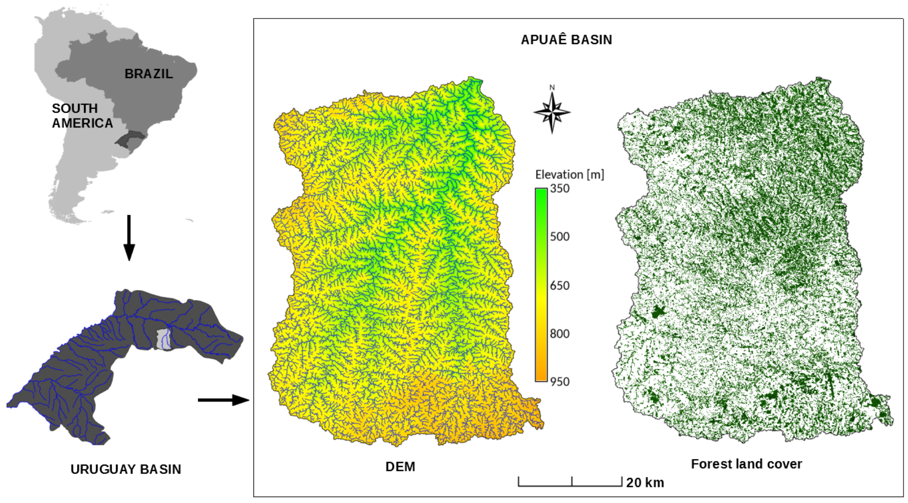


Figure 2.3: Location of the Apuaê basin, Digital Elevation Model (DEM) and forestry maps. Elevations in the basin range from 387 to 929 meters above sea level. Forestry accounts for 36% of the basin area.

Calibration and validation of the model

The calibration of the model considered the case of heterogeneous and anisotropic media. To this end, the soil hydraulic conductivities for the basin were considered to be dependent upon land cover. The land cover was classified into two categories, forest and non-forest, on the basis of previous studies that concluded that hydraulic conductivities have higher values in soils covered by forest [12, 85, 86]. The water outflow from each cell from each time step is summed up and represents the total subsurface flow at the basin outlet. This approach disregards the overland flow travel time in drainage channels, therefore implying it is shorter than the data time step (day). Thereafter, the calculated subsurface flow is compared with the master recession curve. The mean value of hydraulic conductivity in the case of homogeneous and isotropic media was applied to establish the intervals of calibration for the heterogeneous and anisotropic case. The upper and lower bounds for the values of hydraulic conductivity are presented in Table 2.2. The results of the calibration process are shown in the subsequent figures: Fig.2.4 presents the master recession curve for the basin (in red), and was obtained by applying the Matching Strip Method [53, 87] to 29 sets of observed data describing recession events for the basin (in blue); Fig. 2.5 shows the evolution of relative calibration errors; Table 2.3 shows the values of the model parameters obtained from the calibration process.

Table 2.2: Calibration intervals for the hydraulic conductivity.

Land cover class	Parameters	Limits (mday^{-1})
Forest	Upper bound	8.5
	Lower bound	4.3
Non-forest	Upper bound	4.3
	Lower bound	2.1

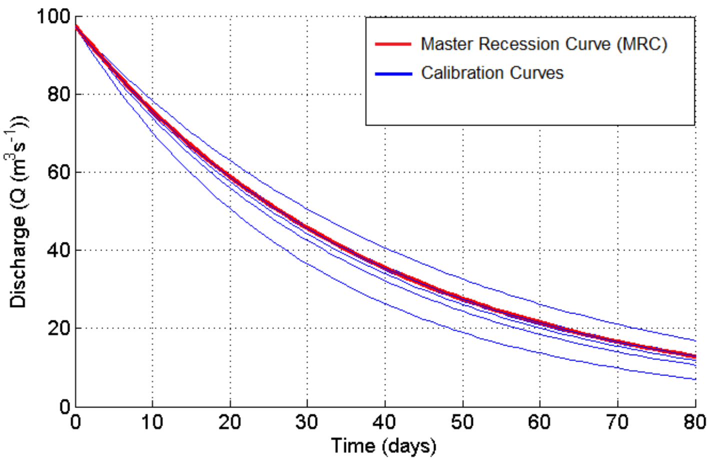


Figure 2.4: Recession curves for the basin resulting from the calibration process.

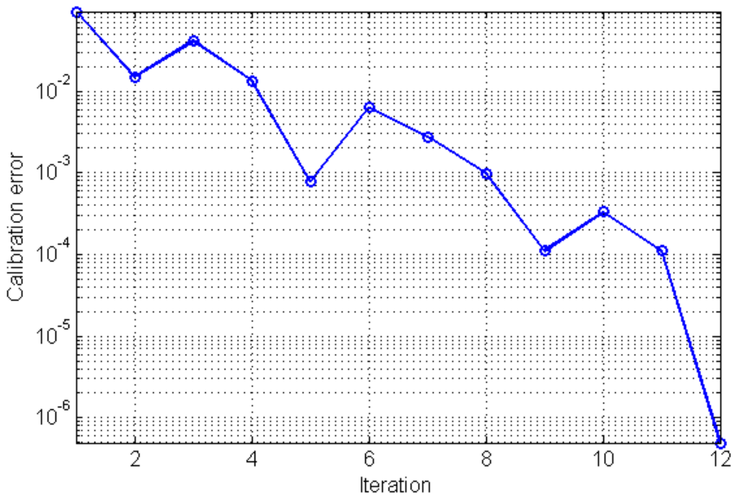


Figure 2.5: Evolution of relative calibration error, which measures the deviation between the actual recession coefficient and that calculated under the conditions of heterogeneous and anisotropic media.

Table 2.3: Parameter values obtained from the calibration process.

Class	Parameter	Value
Forest	K_f	$6.34 \times 10^{-5}\text{ms}^{-1}$
Non-forest	K_{nf}	$3.17 \times 10^{-5}\text{ms}^{-1}$
Homogeneous condition	K	$4.94 \times 10^{-5}\text{ms}^{-1}$
-	Error	4.94×10^{-7}
-	r	-0.025288

The parameter values presented in Table 2.3 correspond to the couple of values K_f, K_{nf} that minimize the distance between the vector of actual discharge and simulated discharge in the drainage basin outlet. The values in Table 2.3 were found to be consistent with characteristic values of hydraulic conductivity from experimental studies that considered the same type of soil and land cover [88, 89].

Simulation of an 80-day drought

The application of the GPGPU-accelerated groundwater flow model for unconfined aquifers is presented in this section. We simulate an 80-day drought period to evaluate the behavior of the aquifer in the area covered by the basin and its impact upon the basin flow. Figure 2.6 presents the variation of soil saturation (expressed by percentages of the basin area) on the topographic surface as a function of time. More intense variation in the soil saturation on the surface is verified during the first 30 days of simulation. This behavior reflects the dependence of the hydraulic conductivity on the thickness of the saturated layer, b, whose value is higher at first and decreases as the aquifer level lowers. The reduction in hydraulic transmissivities in the saturated layer causes a decrease in the value of base flow, whose rate is determined by the recession coefficient, r. The exponential behavior of the discharge can be verified in Figure 2.4.

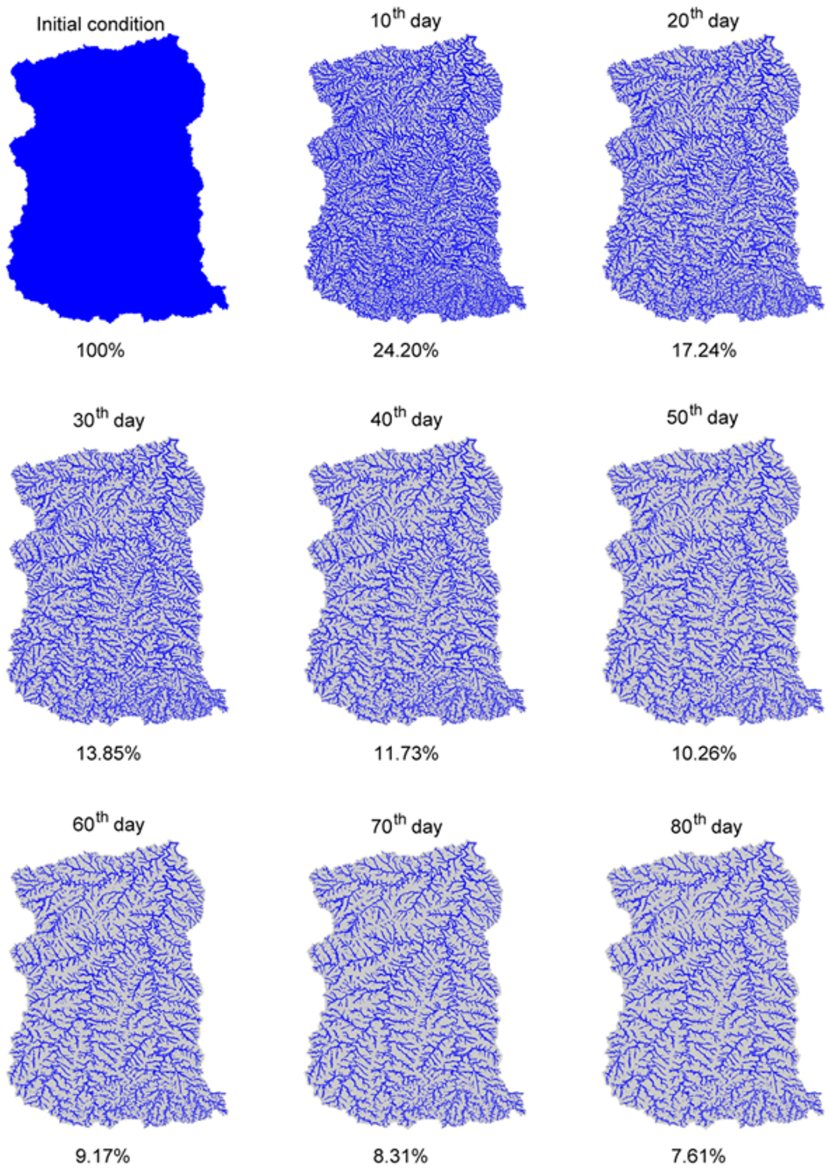


Figure 2.6: Variation in the soil saturation during an 80-day drought. The amount of saturated points in the grid is shown as a percentage at the bottom of each frame.

Figure 2.6 provides a means for the physical interpretation of the process that generates the base flow. It can be noted that the lower portions of the basin topography receive an influx of water from upstream. As the soil saturation level is achieved, the water flows to the drainage channels of the basin, thus generating base flow. Due to the continuous flow of underground water to the drainage channels, the thickness of the saturated layer decreases. As a consequence of the dependence of transmissivities upon the thickness of the saturated layer, the depth of the aquifer increases faster in regions where the saturated layer is thicker. As the process evolves, the water flows downstream until the depth of the aquifer reaches the depth of the confining surface. At such point, the groundwater flow to the river ceases. This description of the behavior of the results based on the simulations agree with the physics of the problem. This indicates that the model captures the essential features of the phenomenon under study.

2.4 Discussion

The GPGPU-accelerated implementation presented in this paper was found to be able to deliver the transient solution for a problem with 5 million unknowns in roughly 4 minutes, as shown in the Results section. This result, along with those cited from the literature, is encouraging as one takes into consideration the increasing demands of scientific computing.

Regarding the solution procedure, the considerable speedup in the parallel code in relation to the serial one can be assigned to the large number of calculations needed to mount the linear system matrix and vectors at each time step. The mounting of the system matrix and vectors is required as the groundwater model for unconfined aquifers means the hydraulic transmissivities are dependent on the thickness of the saturated layer which changes at each time step in the transient model. In fact, it was found that the mounting of the system is the most time-consuming task in the overall solution process. This is an important point to take into account in the design of the solution flowchart, which should consider the parts of the code to be executed in parallel kernels.

Regarding the solution of the groundwater flow model, we note that the relationship between the variations in the depth of aquifers and the fluctuations in the flow rate of water in rivers is an important hydrological issue. Such relationships have significant importance in the analysis of water storage and transport potential in aquifers, as pointed out in Ref. [74]. Such analysis can provide information for the planning of water supply facilities and guidelines for the installation and use of irrigation systems. In a broader sense, they can provide fast and sufficiently accurate scenarios regarding the availability of water in a supply system. Furthermore, they can guide projections and strategies regarding the most favorable time for water authorities to resort to rationing periods. The simulation of groundwater flow for Apuaê basin, Brazil, by means of the GPGPU-accelerated groundwater flow model permitted the assessment of important aspects of its dynamics, while also serving as an illustration of the application of the computational model. In this context, a key point to note is that the transient solution of the groundwater model with over 5 million points was obtained by means of an inexpensive office computer and thus can be largely replicated.

2.5 Final remarks

This study addressed the feasibility of the application of GPGPU to the problem of groundwater flow in unconfined aquifers for anisotropic and heterogeneous media. The results agree with previously published papers in the sense that the parallel implementation largely outperforms the serial one, thus enhancing the computational performance and reducing simulation time. It was found that the CUDA$^{\text{TM}}$ implementation of the groundwater flow equations was capable of outperforming the serial implementation and providing up to a 56-fold reduction in computation time.

With regard to future perspectives, we would like to point out that it is likely that an increasing number of environmental models will have their parallelized version developed and implemented in order to check further advantages and drawbacks of the GPGPU application in the modelling of environmental phenomena. The code for the computational model developed in this research can be obtained at `http://modelagemambientaluffs.blogspot.com.br/` [78].

Chapter 3

Modelling forest regeneration for performance-oriented riparian buffer strips[1]

3.1 Introduction

Numerous scientific studies worldwide have discussed the importance of the riparian buffer strips (RBS) in the conservation of species and habitats, water courses, water quality and quantity [36, 37, 43, 90, 40, 38, 42, 41, 3]. Among the well-established facts in this regard, it is known that RBSs accomplish a number of essential conservation functions, such as filtering of surface and subsurface flow, mitigation of erosion in water bodies, improving microclimate, trapping suspending sediments, adhering and assimilating nutrients, binding dissolved pesticides, among others [37, 40]. The contamination of water by residues of nutrients from agriculture croplands is of special concern. This is due to the fact that nutrient inputs occur systematically along the years, thus having cumulative effects. Indeed, nutrient contamination is one of the main reasons for which the regeneration and maintenance of RBS has become a central argument in favor of water quality. The importance of vegetation resides in the plant uptake and storage of nitrogen, which reduces nitrogen concentration in the discharge flow. As such, the RBS acts like a filter. Nevertheless, this requires that buffer strips have sufficient width and vegetation density, such that they can handle the nutrient load to which they are subjected. While denitrification appears as the only form of permanent nitrogen removal [91, 92, 93], the presence of RBS plays an important role in the process. During the denitrification process, oxidized forms of nitrogen such as nitrates and nitrites are continuously reduced through chemical reactions promoted by the metabolism of specific bacteria to the N2 molecular nitrogen condition that corresponds to the permanent removal of organic nitrogen [94, 95, 92]. As the process must occur under adequate moisture conditions and sufficient amounts of organic carbon, the maintenance of RBS are acknowledged to be favorable for reactions to occur satisfactorily [96]. These conditions are provided by RBS as plant cover allows greater moisture retention and nutrient cycling by continuous degradation of organic material, while it also provides organic carbon

[1]Adapted from [2]

required [91, 97, 98]. The factors that influence degradation have been reported in a variety of researches, such as soil redox potential conditions, temperature, groundwater level, soil permeability, moisture content, successional level and soil cover [91, 97, 99, 100, 101]. Thereby, the maintenance of green areas, especially RBS, as recognized to be fundamental from ecological point of view. In this sense, restoring and preserving RBS areas has a special potential for water quality control through the removal of contaminants such as nitrogen through the improvement of denitrification processing conditions [91, 102]. Although the functions and importance of RBS are widely acknowledged, the degradation of forested areas in the vicinity of water bodies is a recurrent problem worldwide [36, 37, 103, 90, 43, 42, 104]. Thereby, a major source of concern is the long time it takes to reverse a degradation process. While the degradation of large areas of forest can be a matter of months or a few years, a regeneration process requires several years to settle and at least a couple decades to estabilize, to depend on the bioma [105]. This poses a tremendous challenge to decision-makers on the planning and management practices to be applied in each case. In this context, the application of mathematical models can be of help as they allow management practices to be put to test before they are actually applied in the field. Further, model projections can lead to a rather clear idea on what the observed status of the regeneration process should look like at any given time. This allows managers to evaluate the real process against the observed one and test mid-course management practices at any point of the process. A number of studies have addressed forest recovery by means of diverse methodologies. A comprehensive survey on forest recovery with emphasis on the statistical analysis of post-disturbance recovery time across different biomes is presented in Ref. [105]. Among the interest conclusions, it is found that recovery times vary considerably with the geographical location, e.g., forests in Central America and Africa were found to generally recover faster than those in South America and Asia [105]. As another important research tool, one can cite the the Forest Vegetation Simulator (FVS), a decision support tool developed by USDA Forest Service and in use for over 30 years [106]. The FVS encompasses models of forest dynamics that simulate forest vegetation response to management actions, natural succession and disturbances [107, 108]. A recent study of modeling forest growth was presented in Ref. [109] by means of the DLG model. Unlike the FVS, which implements detailed modeling to the tree level, thus requiring detailed inventory of forest and individual trees atributes, the DLG model works solely on the basis of vegetation density over the forested domain of interest. In the case of Ref. [109], the DLG model is applied to the evaluation of forest recovery in Puerto Rico for an interval of 40 years. Besides presenting results of practical interest, the authors point the importance of the DLG model not only in the study of forest recovery, but also to the study of changes in the rate of change and the rate of spread of vegetation over time for a particular location of forested area. This reinforces the importance of adequate model calibration to the local conditions of the region of interest. As an instance, the FVS is reported to have twentytwo variants aimed at accounting for variations in local specificities of forested areas of the United States, British Columbia and Canada [106]. Another instance of modeling tool applied to decision support system is NED (Forest Ecosystem Decision Support Software) [110, 111]. Also developed by the USDA Forest Service, NED is a project-level planning assistance tool with flexible design and intensive requirement for inventory data [110, 111]. Further examples may be found in the literature. In general, forest growth models available in the literature regard individual growth assessment. For example, the AmapSim software presented in Ref. [112] is based on botanical principles and is able to simulate the evolution of individual growth in 3D on the basis of data derived from real plants, taking into account ramifications, bud formation, leaf production, differentiation and formation of lateral shoots and axial growth, among others. The author presents a complete model for growth simulation with support for different species from the knowledge of detailed botanical characteristics and even reports its use to predict fossil species [112]. Other models also follow a similar feature as they focus on individual growth. The work of Twilley et al. [113] presents a numerical model for simulation of vegetation growth in mangrove areas, by means of the so-called FORMAN model. In such model, the conditions of salinity, temperature, nutrient availability and luminosity are considered and growth is described by increasing the diameter-at-the-chest of the individuals. It is worth remarking that the application of this model requires a very detailed inventory of the area to be simulated, since the initial conditions for growth simulation need to be properly assigned for each individual [113]. Some models, such as the Capsis model [114, 115], StoMat (Stochastic matrix models) [116] and SELVA [117, 118] incorporate models of individual growth, the relation of seed dispersal and growth as a function of distance from the source and even the competition among individuals, as in the SILVA simulator [119]. In turn, Porté and Bartelink [120] present an extensive study on forest growth models, raising concern about the limitations of the single-tree growth models commonly adopted. A particularly important point is that the study of large areas by means of single-tree models may consume too much time and effort, since they require detailed tree inventory. In this context, the DLG model [109], appears as an alternative to single-tree models, since it regards forested areas as a whole while it is tailored to describe the time evolution of forest density as a function of diffusion and growth rate parameters [109]. Bottom line, computer models and tools can effectively promote the knowledge-based decision making in the context of forest management and policy making, thus contributing to ecological, economic and social well-being. Regarding forest management practices and their influence upon ecosystem services (ES), recent studies propose the application of models to simulate and optimize objective functions related to desired ES capabilities [121, 122]. Indeed, the use of modeling and programming techniques is reported to allow the identification of feasible solutions to plan for management goals [122]. In the same direction, the application of artificial intelligence techniques has contributed to the assessment of feasible solutions to environmental problems in a wide range of situations [123, 124, 125, 126, 127, 128, 129, 52, 130, 3]. The increasingly large availability of environmental data makes neural networks particularly interesting due to their intrinsic ability to learn from them. Of particular interest in this study, the design of performance-oriented RBS was shown to be feasible as approached from the viewpoint of an Artificial Neural Network Ensemble (ANNE) [3]. As a result, the ANNE was capable of learning from a dataset how to approximate the width of the RBS capable of filtering 90% of the nitrogen intake from uphill agriculture as a function of vegetation cover type (VCT), soil type (ST), mean nitrogen intake (MNI) and removal efficiency (RE) [3]. In this paper, we combine different techniques proposed in the literature (ANNE, DLG) and propose a comprehensive methodology to the study of forest growth from the viewpoint of management and engineering. Towards this, we combine the application of an ANNE to the design of RBS width oriented to performance (as originally proposed in Ref. [3]) to the application of the DLG model of forest growth, as proposed in Ref. [109]. The main contribution of the paper is to propose a comprehensive and practical approach that allows the design of performance-oriented RBS. Following the proposed framework, not only the attributes of the RBS are assessed, but also design requirements can be introduced and a detailed account of its time evolution can be obtained. To illustrate the proposed approach, we apply it in the study of regeneration of RBS in the Ligeiro River watershed, Brazil. The DLG model is calibrated and validated using Landsat images and then applied to simulate vegetation growth up to 30 years ahead. We show that the proposed approach is highly reproducible and, upon further development, can be made an effective design tool.

3.2 Materials and methods

This section addresses the DLG model [109] and its discretization, the procedures for image acquisition and processing and also the procedures for training, validation, test and composition of the ANNE. A flowchart of the methodological procedures is presented in Fig. 3.1.

3.2.1 A model of forest recovery

To describe forest recovery, we consider the diffusive logistic growth model, as presented in Ref. [109]. The model describes forest recovery as a combined process of diffusion and logistic growth. Diffusion is the net flow of particles down a concentration gradient, as described by Fick's law. In the two-dimensional form, for a concentration u, the Fick's lack can be stated as

$$J = -D\nabla \cdot u(t,x,y) \tag{3.1}$$

where J is the amount of substance per unit area per unit time, D is the diffusion coefficient and x,y are the position coordinates, all relative to the concentration $u(t,x,y)$ [131]. In this

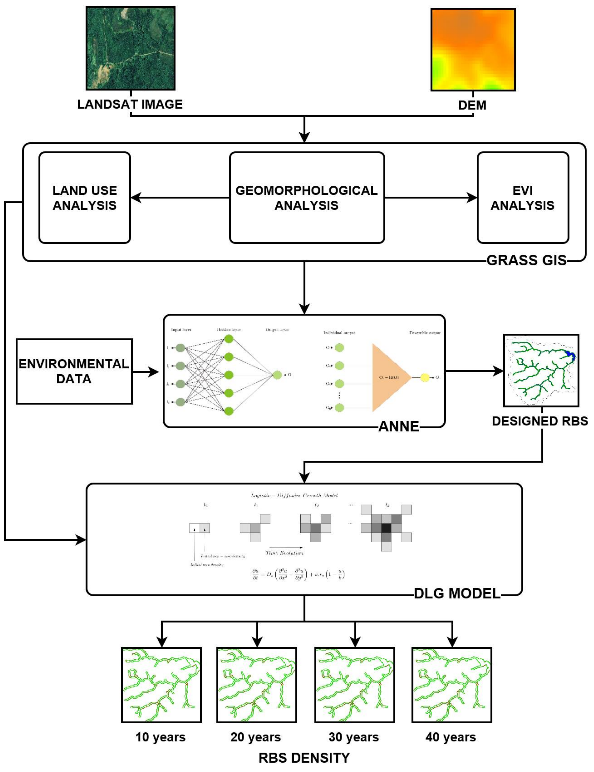


Figure 3.1: Flowchart of the methodological procedures.

formulation, we are assuming that the diffusion coefficient D is homogeneous along the spatial domain and constant over time. This means that the diffusion flux in a position x is proportional to the negative of the gradient of concentration in that position [132, 133, 134]. In turn, the logistic growth model describes population growth subjected to a limitation in the supply of resources, originally proposed by Verhulst [135, 136] and subsequentially applied in the study of several forms of population growth problems, both theoretical and practical [137, 138, 139, 140]. The logistic model states that the rate of change of a population of size U subjected to a constant growth rate r_u in an environment with carrying capacity k_u can be modelled as

$$\frac{\partial u}{\partial t} = r_u u \left(1 - \frac{u}{k_u}\right) \tag{3.2}$$

The solution of the logistic model in Eq. 3.2 can be found analytically and it is given by

$$u(t) = \frac{k_u u_0 e^{r_u t}}{k_u + u_0 \left(e^{r_u t} - 1\right)} \tag{3.3}$$

and from Eq. 3.3 we note that, for any initial condition on the population size or density, the population converges to the carrying capacity as time tends to large values. This can be expressed more formally as

$$\lim_{t \to +\infty} u(t) = k_u \tag{3.4}$$

which indicates that $u(t) = k_u$ as $t \to +\infty$. Without loss of generality, we consider $U(t,x,y)$ as a density variable whose values lie in the interval $[0,1]$.

Consider the continuity equation in two dimensions for the quantity u, given as

$$\frac{\partial u}{\partial t} = -\nabla \cdot J + q \tag{3.5}$$

where $q = q(t,x,y)$ is a source or sink term. Substituting Eq. 3.1 and Eq. 3.2 in Eq. 3.5 and considering $q(t,x,y)$ to behave as the logistic equation for $k_u = 1$, one comes to

$$\frac{\partial u}{\partial t} = D_u \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + r_u u(1-u) \tag{3.6}$$

Equation 3.6 is named the Diffusive Logistic model [141, 134, 109]. The illustration of logistic growth and diffusive-logistic growth is presented in Fig. 3.2.We follow Ref. [109] and apply the Crank-Nicolson scheme of Finite Differences [67] to evaluate the model numerically. The boudaries of the RBS region are assumed to be Neuman, with $\partial u = 0$, that is, a no-flux condition.

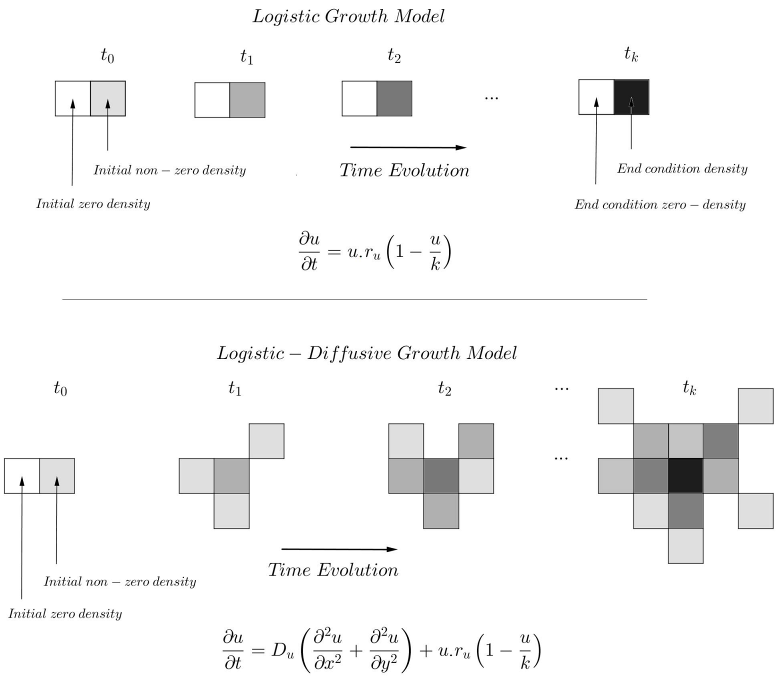

Logistic Growth Model
t_0
t_1
t_2
...
t_k
Initial non − zero density
Initial zero density
Time Evolution
End condition density
End condition zero − density
$\frac{\partial u}{\partial t} = u.r_u\left(1 - \frac{u}{k}\right)$
Logistic − Diffusive Growth Model
t_0
t_1
t_2
...
t_k
Initial non − zero density
Initial zero density
Time Evolution
$\frac{\partial u}{\partial t} = D_u\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + u.r_u\left(1 - \frac{u}{k}\right)$

Figure 3.2: Illustration of logistic growth (above) and diffusive-logistic growth (below).

3.2.2 Discretization and solution of the DLG model

We follow the procedure in Ref. [109] and discretize the DLG model using the Crank-Nicolson scheme [67]. The time derivative in Eq. 3.6 is approximated with first-order forward difference, that is,

$$\frac{\partial u}{\partial t} \approx \frac{u(x,y,t+\Delta t) - u(x,y,t)}{\Delta t} \tag{3.7}$$

where Δt is the time step. The spatial derivatives are approximated with second-order differences

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u(x+\Delta x,y,t) - 2u(x,y,t) + u(x-\Delta x,y,t)}{(\Delta x)^2} \tag{3.8}$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{u(x,y+\Delta y,t) - 2u(x,y,t) + u(x,y-\Delta y,t)}{(\Delta y)^2} \tag{3.9}$$

We assume the parameters D_u and r_u are constant over time and space and the carrying capacity is $k_u = 1$. Taking the finite difference approximations from Eqs. 3.7, 3.8 and 3.9 to Eq. 3.6 and expressing the value of the continuous function $u(x,y,t)$ in a point $\left(x_i, y_j\right)$ of the grid at time step t_k as U_{ij}^k, one obtains an expression for the finite-difference approximation of Eq. 3.6. After some algebraic manipulations and assuming $\Delta x = \Delta y$, and defining $\lambda = \frac{\Delta t D_u}{2 \triangle x^2}$, it can be written as

$$\begin{aligned}(1+4\lambda)U_{i,j}^{k+1} - \lambda\left(U_{i+1,,j}^{k+1} + U_{i-1,,j}^{k+1} + U_{i,,j+1}^{k+1} + U_{i,,j-1}^{k+1}\right) = \\ \cdots = \left(1 - 4\lambda + r_u\left(1 - U_{i,j}^k\right)\right)U_{i,j}^k \\ +\lambda\left(U_{i+1,,j}^k + U_{i-1,,j}^k + U_{i,,j+1}^k + U_{i,,j-1}^k\right)\end{aligned} \tag{3.10}$$

By rewriting Eq. 3.10 in compact format as a matrix equation, it reads

$$AU^{k+1} = B^k U^k \tag{3.11}$$

The solution U^k for the time step k is known and this makes BU^k known. We define $b^k = BU^k$ to express the linear system in the usual notation $AU^{k+1} = b^k$. The formulation of the problem guarantees that, for each time step k, the linear system in Eq. 3.11 has a unique solution vector U^{k+1}, which can be expressed as

$$U^{k+1} = A^{-1} b^k \tag{3.12}$$

Now, A is a constant sparse and symmetric matrix, and as the system has to be solved once at every time step (not included the initial condition U_0).

We solved the resulting linear system in Eq. 3.11 using the Stabilized Bi-Conjugate method with error tolerance $\varepsilon = 10^{-9}$ for each time step. The solution was implemented and computed in GPGPU (General Purpose Graphics Processing Unit) using CUDA (Computer Unified Device Architecture) parallel programming and computing platform [142, 57] and the CUSP library [68]. To this end, we employed an NVIDIA(TM) GeForce(TM) GTX 1060 graphics card device along with an Intel(R) Core (TM) $i7-7700$ host processor.

3.2.3 Image acquisition and processing

The Landsat 5 and 8 scenes were acquired from the US Geological Survey [4, 5, 6] and converted from Digital Number (DN) to reflectance in the software GRASS GIS 7.0 using the module *[i.landsat.toar]* [143]. The scenes from the Landsat 5 TM are from the years 2000 and 2010, path/row=222/79, DOY = 38 and 49, respectively. The scene for the year 2016 is from the OLI/TIRS sensor, path/row = 222/79 and DOY=18. The resulting image was converted to vegetation density by means of the implementation of Enhanced Vegetation Index (EVI) in module *[i.vi]* [143], and then saved as a matrix of numeric entries correponding to the vegetation density in each pixel of the image. This procedure was performed for each of the three scenes to serve as initial/final conditions for calibration/validation of the model and as initial condition for application in the assessment of RBS regeneration in the Ligeiro River watershed.

3.2.4 Calculation of vegetation density

The vegetation density for the calibration, validation and application of the model was calculated by means of the Enhanced Vegetation Index (EVI) [144]. One principe behind the EVI is the estimation of atmospheric effects by means of the difference between red and blue band reflectances, which scales with the concentration of aerosols [144]. Another principle is the reduction of distortions of light reflected by ground cover below the vegetation, which influence the reflectances in the red and near infra-red bands [144]. Thus, the EVI seeks the optimization of vegetation signal by intensifying the response in regions with concentration of biomass. The equation for vegetation density in an image pixel is given by

$$EVI = G\frac{NIR - RED}{NIR + C_1 \cdot RED - C_2 \cdot BLUE + L} \tag{3.13}$$

where G is a gain scaling factor, C_1 and C_2 are aerosol resistance coefficients, L is the canopy adjustment parameter and NIR, RED, $BLUE$ are reflectance in the near-infrared, red and blue spectral bands, respectively. In this paper, the EVI in Eq. 3.13 is applied to Landsat images in order to obtain the initial conditions to the calibration, validation and application of the model. We apply the standard parameter values $G = 2.5$, $C_1 = 6$, $C_2 = 7.5$ and $L = 1$ [144]. The initial and final conditions for calibration/validation of the model were obtained from Landsat scenes, respectively in Refs. [4, 5].

3.2.5 Calibration and validation of the DLG model

The calibration of the model is aimed at identifying the couple of parameters D_u, r_u that minimize the Root Mean Squared Error (RMSE). To this end, a systematic search was performed over the parameter space $D_u \times r_u$, starting from coarser grid resolutions and successively increasing the resolution around points of minimum from coarser levels. The calibration and validation were performed considering an area of permanent conservation with the same type and pattern of vegetation as the region of interest. In the calibration process, we used the half of the image of vegetation density, while the other half was saved for validation. The validation process was also performed using the RMSE.

3.2.6 Artificial Neural Network Ensemble (ANNE) and the estimation of RBS width for the filtering of nitrogen

The application of an ANNE to the estimation of RBS width was proposed in Ref. [3]. We present a summary of the key points that are relevant to the methodology proposed in this paper.

The training, validation and test data

The training, validation and test of the neural networks were performed with data originally published in a number of independent studies which were collected and reproduced in Ref. [90]. The data include mean nitrogen influent (MNI, in ppm), mean nitrogen effluent (MNE, in ppm), removal effectiveness (RE, in %), vegetative cover type (VCT, classified into grass, grass forest or forest), soil type (ST, classified into sand, silt, clay and combinations) and buffer width (BW, in meters). These indicators are associated to the mitigation potential of non-point source nutrient pollution. In turn, nutrient pollution is mainly associated to the accumulation due to the loss of nitrates from upslope agriculture areas. The dataset from Ref. [90] provided 39 complete records from which the amounts of nitrate and total N loads was associated with the indicators MNI, RE, VCT and ST. Such indicators were used as inputs to the neural network while BW was used as output. The sigmoid function was chosen as activation function of the neuron. To scale the data to region of interest of the sigmoid function under this application, the data were linearly mapped to the interval $[0,1]$. Thus, the number of inputs to each ANN in the ensemble is $N_I = 4$ and the number of output is $N_O = 1$. The input variable ST was converted to their characteristic hydraulic conductivity coefficients as found in the literature [89]. In turn, vegetation types were converted to density parameters in the interval $[0,1]$.

The application data

The geomorphological information of the study area was obtained from the digital elevation model (DEM) [145, 146]. The DEM was processed using the open source software GRASS GIS 7.0 [143]. The raster maps were derived by means of the module *r.watershed* and then the watershed slopes, drainage directions, sub-basins and stream segments were obtained. A minimum sub-basin area threshold was chosen to subdivide the watershed such that every sub-basin contributes for at least one stretch of river. The stream segments were delineated by means of a calibrated channel initialization threshold which defines where a river begins [147, 148]. The delineated streams were compared to the location of river heads identified by satellite images in order to check the results. The VCT for the region of interest was obtained by means of image processing and *in loco* validation of a dozen key points. The scene (Landsat 8 scene in Ref. [6]) was obtained from the US Geological Survey [149]. The land use classification was performed using the unsupervised cluster analysis algorithm for pixel grouping of the watershed land use in 20 classes. Such classes were grouped into 3 classes of vegetation cover type based on vegetation density. Each class was assigned an index according to vegetation index, resulting in classes 0.7 (forest), 0.5 (grass forest) and 0.3 (grass). Next, the MNI for each sub-basin was estimated by means of experimental data from soybean croplands in the literature [150]. The MNI was obtained under the hypothesis that the contribution of each sub-basin scales with its cropland area and concentrates equally along the length of the stretch of river to which it contributes for a constant depth. A 20-year time series with daily rainfall data were obtained from eight rainfall gauges, while river level was obtained from one river level gauge from the National Agency of Water Resources (ANA) website [151]. The Inverse Distance Squared Weighting (IDW) algorithm was applied to interpolate the rainfall data within the watershed. The amount of evapotranspiration was estimated by means of water balance calculation for the period. As a result, the average annual rainfall, evapotranspiration and runoff were obtained as 2210.6 mm, 1247.5 mm and 963.1 mm, respectively. As the major interest resides in the sub-superficial transport that takes places in the hillslopes, we regard the amount of precipitation that generates runoff. As a result, the estimated MNI for the $j-th$ sub-basin was estimated as

$$MNI_j = \frac{1}{\mu} N_j^{basis} \cdot \frac{A_j^{cropland}}{L_j^{river}} \cdot \frac{1}{P_j^{runoff}} \tag{3.14}$$

where MNI_j (ppm) is the mean nitrogen influent for the $j-th$ sub-basin, N_j^{basis} $(mg\,m^{-2})$ is the typical nitrogen load for the respective cropland, μ is the soil porosity, $L_j^{river}(m^2)$ is the area of the longitudinal section of the river that receives contribution of the $j-th$ sub-basin, $A_j^{cropland}$ (m^2) is the area of the $j-th$ cropland, P_j^{runoff} $(l\,m^{-2})$ is runoff. Considering the river morphology, it is assumed that all longitudinal sections of the river have unitary depth.

Architecture, training and validation

The choice of the network architectures was based on the upper and lower limit equations for the number of neurons in the hidden layer [152]. According to [152], the number of hidden layer neurons that ensures that the neural network is able to approximate any continuous function is given by $N_H \leq 2N_I + 1$, where N_H is the number of neurons in the hidden layer and N_I is the number of inputs. Further, it is known from the work in Ref. [152] that the number of neurons in the hidden layer must observe the upper limit $N_H \leq \frac{N_R}{N_I+1}$, where N_R is the number of records in the dataset. From the dataset, $N_I = 4$, $N_R = 39$ and then we consider architectures featuring N_H in the range $2 \leq N_H \leq 7$ in order to satisfy both restrictions. Thus, the network architectures considered were 4-2-1 to 4-7-1. We refer to Figure 3.3-(a) for an example of a 4-5-1 architecture). Figure 3.3-(b) shows the process of obtaining the ANNE output from the outputs of the individual ANNs. The neuron activation function was chosen to be the logistic function (see ref. [153]).

Test and selection

The test of the individual ANNs was performed by evaluating their response to unseen inputs from the test set. The individual networks with smallest output error levels were selected to compose the ANNE. The number of ANNs to compose the ensemble was established such that the largest error would not overcome 20%. Further, an ANN with small error would be admitted into the ANNE if and only if its error is uncorrelated or loosely correlated to the errors of previously chosen members of the ensemble. Towards that, the Pearson correlation coefficient was applied to evaluate the correlation in output error across different individual ANNs. Following these criteria, 10 individual ANNs were selected to be part of the ANNE.

Application of the ANNE

The ANNE was applied to design a RBS to the Ligeiro River watershed on the basis of the input parameters of each sub-basin of the watershed. The input parameter RE was set to 90% and this can be regarded as a design criterion. It could be that the RE is set to different values according to the filtering effectiveness required for a particular stretch of river, i.e., nearby a water catchment location, for example. Vegetation density in the watershed was evaluated by means of land use classification in software GRASS GIS 7.0. The soil hydraulic conductivity was taken as $14.6 cm day^{-1}$, a typical value for the Brown latosol in the region according to Ref. [89].

For further details on calibration results, test results and sensitivity analysis of the ANNE, we refer to [3].

3.2.7 Application of the model

The model is applied to the RBS belonging to the Ligeiro River watershed. Since the picture element size of the Landsat image is 30 x 30 meters, we round up the results of the application of the ANNE to the next integer that is multiple of 30 meters, and this will be considered to be the buffer width. To obtain the integration domain, we apply a buffer around the river courses pointed by the digital terrain model (DTM) in GRASS GIS 7.0. The model is simulated to 10,

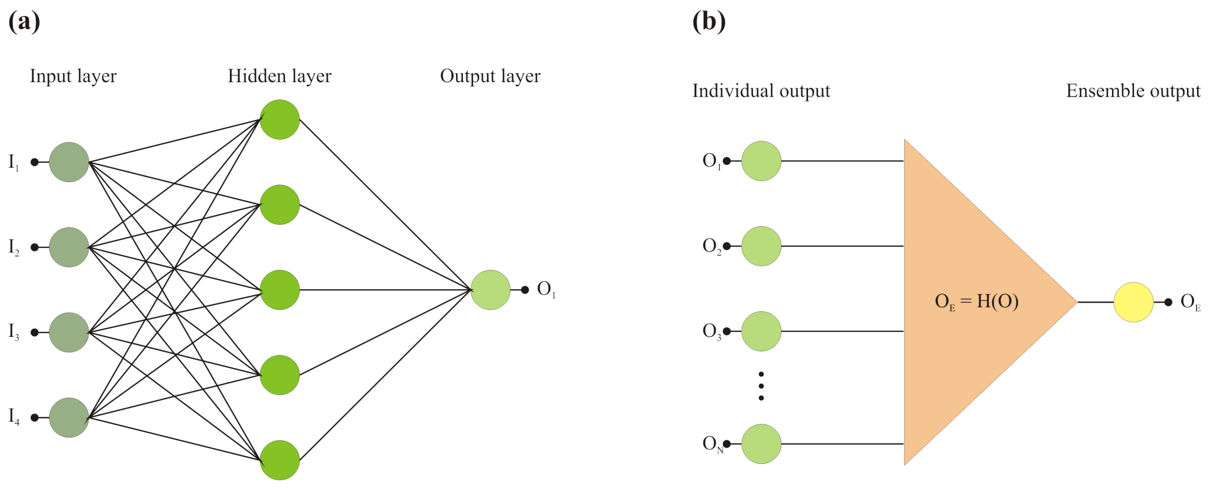


Figure 3.3: (a) Example of ANN architecture featuring an Input-Hidden-Output relation 4-5-1 (four input variables, five hidden nodes and one output variable). (b) Relation between the individual ANN output and the ANNE output. Extracted from [3]

20, 30 and 40 years ahead from the initial conditions U_0 obtained by means of application of EVI to the Landsat scene in Ref. [6].

3.2.8 Strengths and limitations of the framework

Artificial Neural Network Ensembles' responses to the features and requisites of riparian buffer strips

It is widely recognized that artificial neural networks can give very accurate responses when adequately trained, validated and tested [154, 125, 155, 127, 123, 156, 52, 157]. To this end, the data fed to the network must be as representative as possible of the phenomenon to be black-box modeled. As such, the approach presented in this paper, as based on ANNEs, inherits these qualities and premises. In the presence of enough data and adequate training, validation and testing, the RBS features as provided by the ANNEs can give a trustworty approximation, especially to problems where conceptual models cannot be easily applied [3].

The application of Enhanced Vegetation Index (EVI) in Landsat satellite images

The application of EVI to assess vegetation density comes as an alternative to the land-use classification, as proposed in Ref. [109]. The advantages of the application of EVI (or any other vegetation index) are that (i) it results in a pixel-by-pixel vegetation density and (ii) it can be readily scaled to the range of the forest density variable $u(t,x,y)$ in the diffusive-logistic model. Further, vegetation indices are better indicators of lushness and health of the vegetative cover as pictured by Landsat images.

GPGPU implementation of the diffusive-logistic model

The solution of the DLG model was referred to as computationally demanding in Ref. [109], due to the large number of points of the resulting mesh (presumably about 4 million). In order to tackle this issue, the authors redefinined the original grid cells (30 x 30m) calculating the percent cover of each land cover class to generate 1-km grid cells, thus obtaining a mesh of 33 x 139 grid cells (see Ref. [109]). In order to keep the original resolution of 30 x 30 meters, which can provide a better solution for the model, we workaround the high computational demand by implementing a GPGPU solution to the problem using CUDA/C. Through this implementation, we perceived an approximate 50-fold reduction in computational time relatively to the serial CPU implementation.

Solution of the irregular mesh defined in a watershed

The spatial pattern occupied by a river in a watershed is known to be highly irregular, being frequently referred to have fractal structures [158, 159]. Under these conditions, the definition of boundaries conditions for diffusion problems becomes a logistically complex task, as observed in Ref. [109]. To bypass this difficulty, we apply the ANNE solution of RBS width and the Digital Elevation Map of the watershed as inputs and automatize the generation of the corresponding linear system by means of a computer routine. The routine was designed to read the mesh points from the input matrices to define the RBS of interest for every stretch of river and then to apply the no-flux boundary condition all over the mesh. As a result of the automation of the task, we hope that the methodology proposed in this paper can find more widespread applicability.

3.3 Case study: simulation of forest regeneration in the Ligeiro river watershed, Brazil

3.3.1 Characterization of the watershed

The Ligeiro River watershed is situated in the city of Erechim, Rio Grande do Sul state, Brazil, between $27^o39'$ S and $27^o43'$ S, and $52^o14'$ W and $52^o18'$ W. It is $21.18\,km^2$ in area and its elevations above the sea level range from 661 m to 815 m, approximately. The terrain slopes vary between 0.02 and 0.22. Soils throughout the watershed are classified as Brown latosol based on the color of B horizon [88]. The Ligeiro River watershed is located in a region of Atlantic Rainforest, a bioma in which the vegetation is composed primarily of mixed ombrophile forest [160, 161], although its current situation has suffered severe influence of anthropic activities, from which we highlight agriculture [162]. For natural conditions of forest region, the vegetation is characterized by the presence of large trees with species that form canopies of deciduous, semi deciduous and evergreen trees during the winter [163, 164]. The tipical condition of forest is the perennial trees, a fraction of which lose all or part of leaves in the cold seasons. The trees stand out for having long life, reaching live centuries. These forests are most known for harboring the Araucaria angustifolia - the Brazilian pine species which is subject to intensive research, which raises concerns about its conservation [165, 166, 167] as well as being responsible for the conspicuous characteristic of the plant formation to which it belongs [164, 168], is income and food source for local people and wildlife in general. Land use analysis has shown that $4.94\,km^2$ of the watershed area are identified as forests and $16.04\,km^2$ as crop lands. According to the Brazil's Forest Code, the watershed should be $2.10\,km^2$ in area of riparian vegetation (taking into account thirty meters along the river length and fifty meters around every head river). However, it is only $0.94\,km^2$. Moreover, the watershed has been subjected to drought events, which compromised the water supply and raised concerns from water resource managers and society as well. The elevation map of the watershed and the map of land use are shown in Figures 3.4 and 3.5, respectively. The current situation of the riparian vegetation in the watershed is shown in Figure 3.5. These soils constitute the largest class of territorial expression and agricultural potential of the country, being explored with various crops, reforestation and pasture [83].

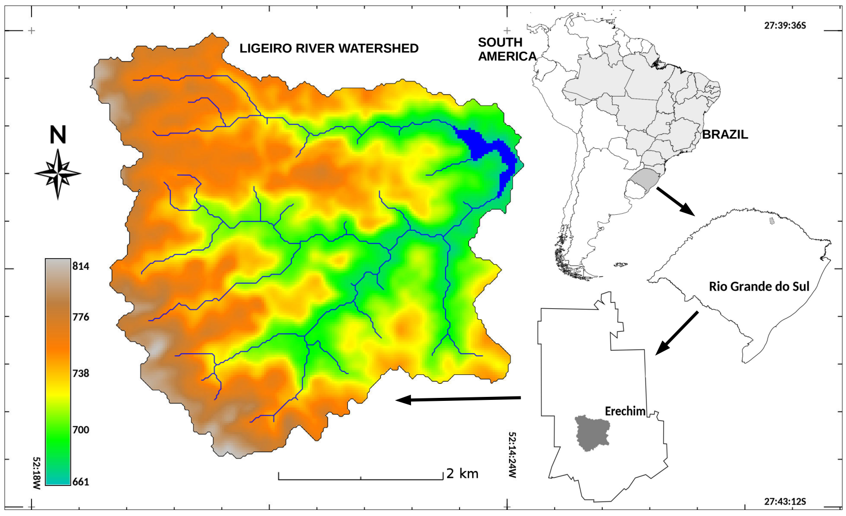


Figure 3.4: Location of the Ligeiro River watershed in Erechim, Brazil, and elevation map showing the drainage network and the water supply reservoir. Elevations in the watershed range from 661 to 815 meters. Extracted from [3].

3.3.2 ANNE and the design of performance-oriented buffer zones: the case of nitrogen filtering

We base our study in the results presented in Ref. [3], where an ANNE was trained, validated and tested to design a RBS to the Ligeiro River watershed. The main design requirement is that

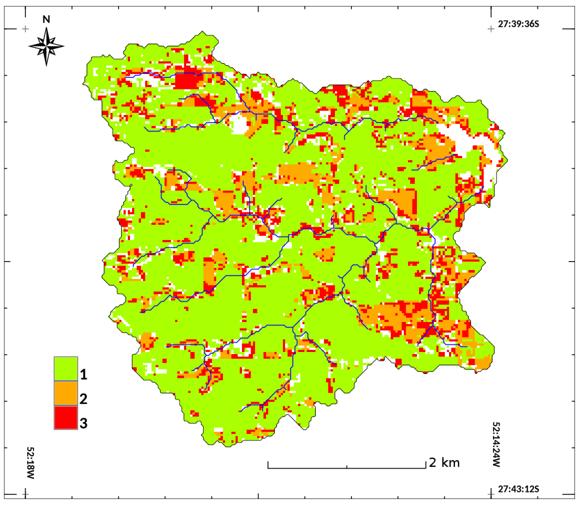


Figure 3.5: Map of land use in the Ligeiro River watershed: (1) cropland, (2) intermediate-density vegetation - grass-forest and (3) dense vegetation - forest. Extracted from Ref. [3].

the RBS should be capable of filtering at least 90% of the MNI. The reason why we set RE to 90% was the characteristic nitrogen load of agricultural catchments for the watershed in which the study took place. Moreover, the RE value might be set up in order to comply with environmental standards for residual nitrogen limits in the river at the target basin. Diffuse sources of pollution represent a significant fraction of the pollution of watercourses, and unlike point sources that can be well defined and measured, these are generally estimated [169]. Each country has, according to its control mechanisms and desired quality, standard concentration values of the contaminants it considers. A simple way to define removal efficiency is to base it on standard values according to legislation. In the case of the basin under study, by Brazilian laws, maximum concentration values are defined according to the classification of the water body [170]. But these legal standards are not always based on the natural ability of self-purification and assimilation as they may be set to values that are not actually attainable. A study that gathers an analysis on the assimilation and removal capacity of nitrogen and phosphorus from diffuse sources in riparian zones in different types of basins by means of the analysis of numerous published works, reveals that the maximum efficiency varies according to the vegetative coverage, succession stage, type of land use [37]. In this way, it seems more adequate to define maximum efficiency according to the intrinsic capacity of the type of plant formation. Nitrogen removal efficiency values for forest riparian zones range from 82 to 87% [171, 172, 173] in older forest areas up to 95% or more in young and growing forests [173, 172, 174, 175, 176]. Thereby, a sensible value for RE according to literature review is about 90% as verified in Ref. [174]. As a result, the adequate RBS width for every stretch of river in the watershed was calculated. The basin was subdivided into 165 sub-basins, each one contributing to a stretch of river. Notably, buffer widths presented values distinguished in classes, despite the variability in the mean nitrogen influent for each sub-basin [3], a behavior that agrees with previous studies [37], as the output of nitrogen and phosphorus into streams was comparably low despite the different input load. This illustrates the fact that the uphill part of the buffer vegetation retains a larger share of the nutrients, since the specific removal per meter is reported to decrease downhill. It is worth noting that the largest width value from the ANNE response, about 47 meters, is rather close to the $50-60$ meter range indicated by the authors in [37]. In fact, the variability in ANNE buffer width was observed to depend more heavily upon the vegetation cover type (VCT) and desired removal effectiveness (RE). Regarding VCT, it was observed in [37] that higher nitrogen uptake observed in younger vegetation makes it more effective in nitrogen removal. This agrees with the results of the ANNE, since it was observed that grass-forest regions were identified as the areas less demanding of buffer width for the same filtering effectiveness. From the training, validation and test, it was observed that the ANNs, and thus the ANNE, could capture the effect of VCT. In this application, we use Landsat 5 images with resolution of 30 meters, such that we approximate the RBS width to the closest integer multiple, 60 meters.

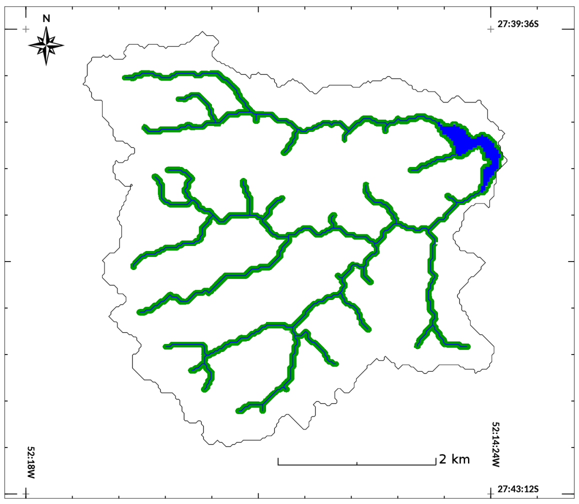


Figure 3.6: Riparian buffer strip widths along the water bodies according to the ANNE. Extracted from [3].

Recalling the results presented in Ref. [3], the total vegetated area demanded by the ANNE for the filtering of 90% of the nitrogen in the watershed amounts $2.91\,km^2$, well above the current vegetated area and even above the legal demand. As we consider the current watershed land use presented in Figure 3.5 and compare to the output of the ANNE, one obtains a picture of the RBS deficit. Indeed, the results indicate that there is a major area of missing vegetation that amounts to $1.97\,km^2$. An important and interesting fact is that the area of missing vegetation according to the ANNE is considerably larger ($+0.81\,km^2$) than would be required by the current environmental law in Brazil ($1.16\,km^2$).

3.3.3 Vegetation density for initial/final conditions

The initial and final conditions for the calibration of the model were obtained by means of the application of Enhanced Vegetation Index (EVI) to the Landsat scenes in Refs. [4, 5]. The results are shown in Fig. 3.7.

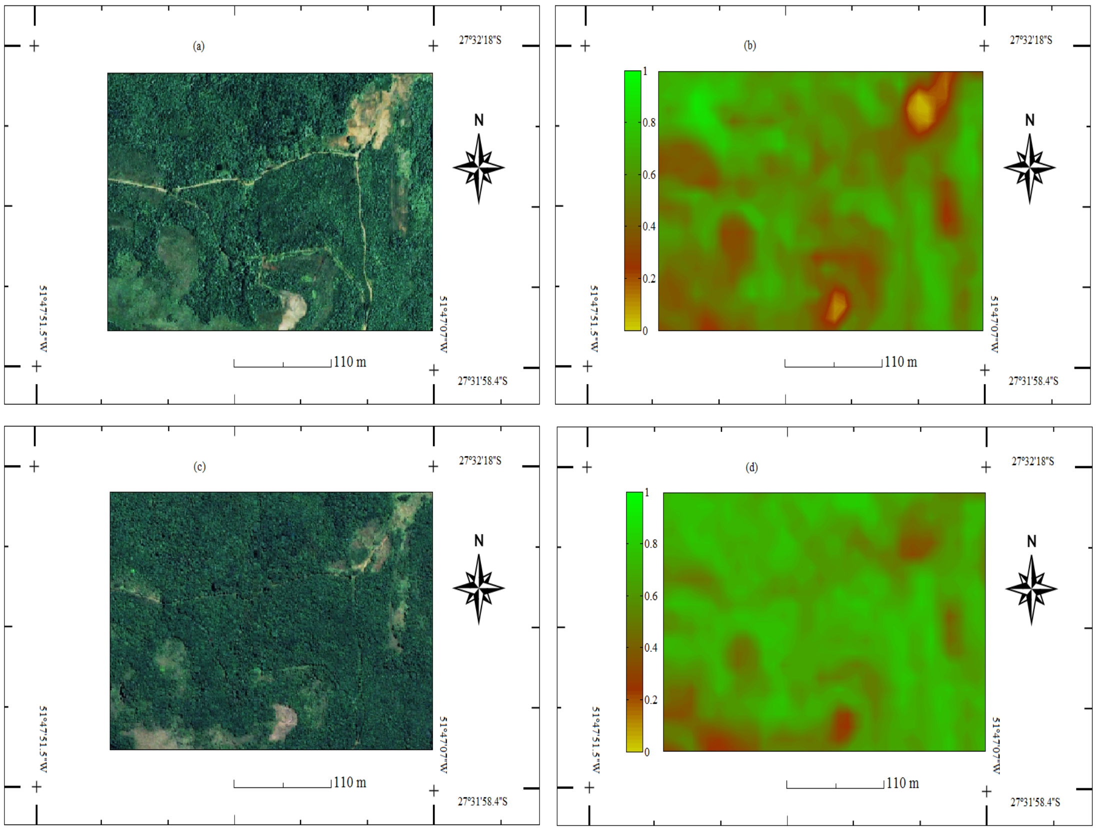


Figure 3.7: Results of the application of Enhanced Vegetation Index (EVI) to acquire initial and final conditions for the model: (a) original high-resolution true color image in the year 2000 and (b) the corresponding EVI vegetation density in the year 2000 [4]; (c) original high-resolution true color image in the year 2010 and (d) the corresponding EVI vegetation density in the year 2010 [5].

3.3.4 Calibration and validation of the model

The model was calibrated and validated on the basis of two Landsat 5 images. The images display a portion of the riparian preservation area surrounding Machadinho dam reservoir and they show the same region in 2000 and 2010 for the same month of the year [4, 5] scenes. For calibration and validation of the model, we divide the images in two and use the left-half for calibration and the right-half for validation. The calibration process searches for the couple of parameters D_u, r_u that minimizes the error between the real and simulated final conditions on vegetation densities. The search was conducted over the parameter space $D_u \times r_u$ by successively increasing the resolution around points leading to minima from coarser levels of grid resolution. In the end of the process, the validation and calibration errors calculated by means of RMSE, were $e_{cal} = 0.080$ and $e_{val} = 0.097$. The results from calibration and validation are shown in Figs. 3.8 and 3.9, respectively.

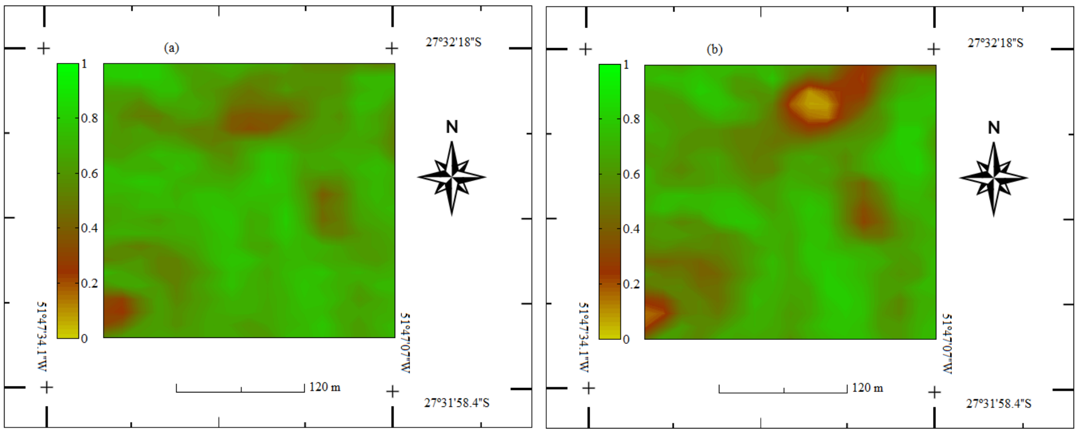


Figure 3.8: Maps obtained during the model calibration process: (a) vegetation density obtained from application of EVI to the (right half of the) Landsat 5 image of the riparian forested area in the surroundings of Machadinho Dam, in the year 2010 [5]; (b) vegetation density obtained by means of numerical simulation of the DLG model, featuring the parameters $r_u = 0.03$ and $D_u = 1.02 \times 10^{-6}$.

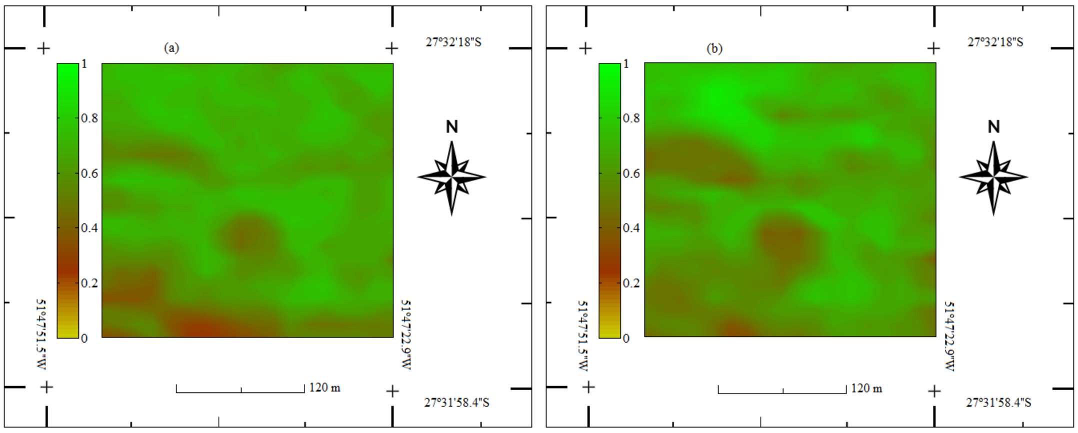


Figure 3.9: Maps obtained during the model validation process: (a) vegetation density obtained from application of EVI to the (left half of the) Landsat 5 image of the riparian forested area in the surroundings of Machadinho Dam, in the year 2010 [5]; (b) vegetation density obtained by means of numerical simulation of the DLG model, featuring the parameters $r_u = 0.03$ and $D_u = 1.02 \times 10^{-6}$.

3.3.5 Application of the model

The model was applied to simulate forest regeneration 40 years ahead in the region of the Ligeiro River watershed. To define the width of the RBS for each stretch of river in the watershed, we considered the results obtained from the ANNE in Subsection 3.3.2. The initial conditions were derived from the application of EVI to the vegetation belonging to the RBS applied to a satellite image from the watershed [6]. The actual mesh was composed of 4,300 points. The initial conditions of the simulation are presented in Fig. 3.10, while the results for 10 years, 20, 30 and 40 years are shown in Fig. 3.11. It can be observed from Fig. 3.11-(d) that, except for some few spots, the RBS is expected to be in an advanced stage of regeneration. The largest growth rate was observed in the first 10 years. The evolution of forest density along the period can also be seen in the density histograms in Fig 3.12.

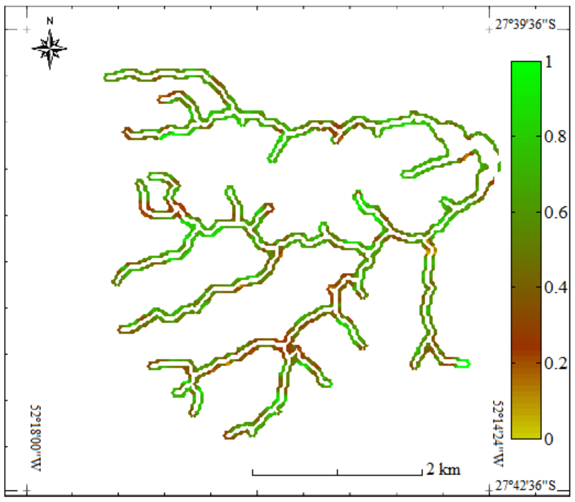


Figure 3.10: Current condition of forest density in the RBS of Ligeiro River watershed, obtained by means of application of EVI to the Landsat scene in Ref. [6].

3.3.6 Sensitivity analysis

Sensitivity analysis of the model with respect to the parameters showed that the diffusive-logistic growth model is most sensitive to the growth rate paremeter, r_u. The results of sensitivity analysis in Fig. 3.13 also reveals that the model can handle uncertainty of about 10% in the value of r_u around the calibrated value without significant changes in the value of RMSE.

3.4 Discussion

3.4.1 Land management and ecology

Forest regeneration is a long-term process whose outcome depends in a considerable extent on the management techniques to be employed. The fact that management strategies can be

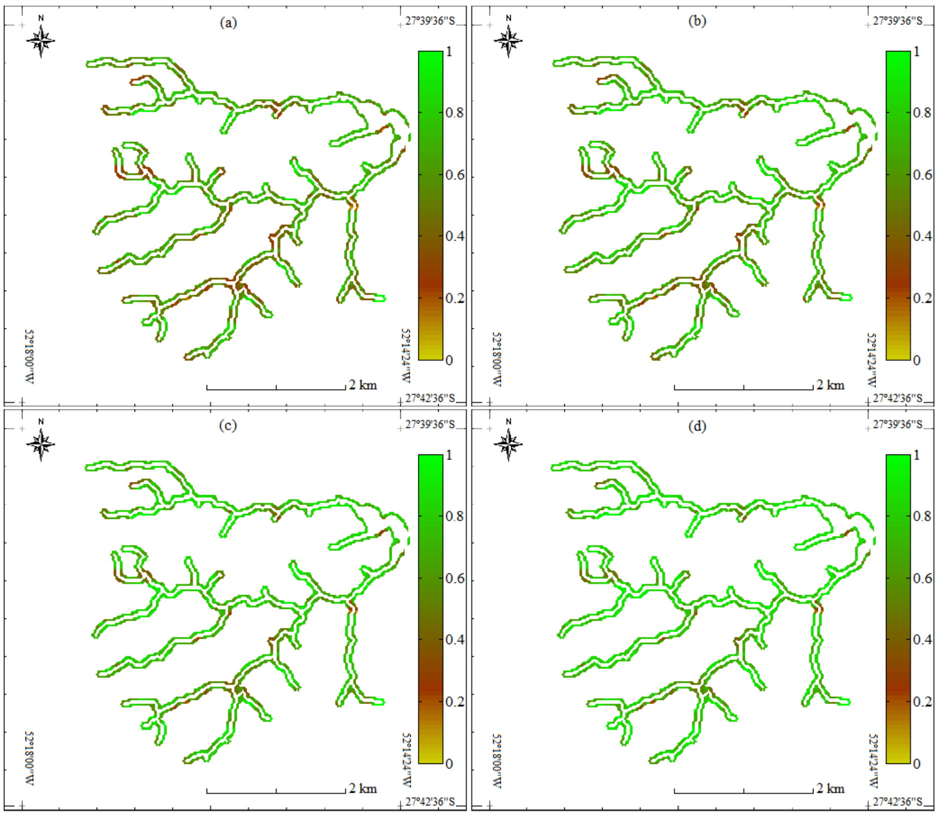


Figure 3.11: Simulation results to forest density in: (a) 10 years, (b) 20 years, (c) 30 years and (d) 40 years to RBS in the Ligeiro River watershed.

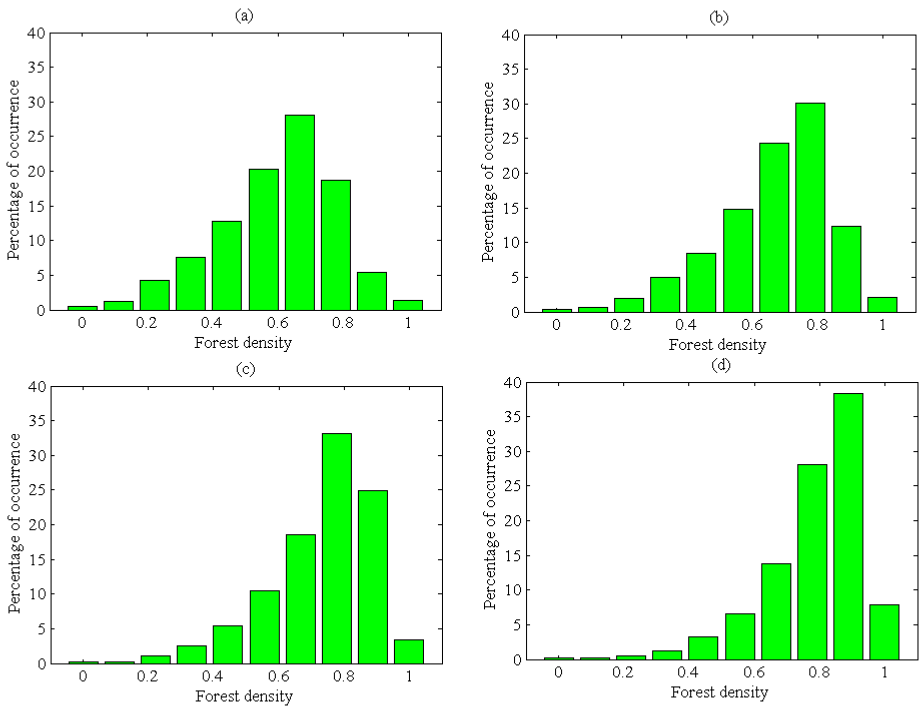


Figure 3.12: Histograms of forest density resulting from numerical simulations: (a) 10 years ahead; (b) 20 years ahead; (c) 30 years ahead and (d) 40 years ahead.

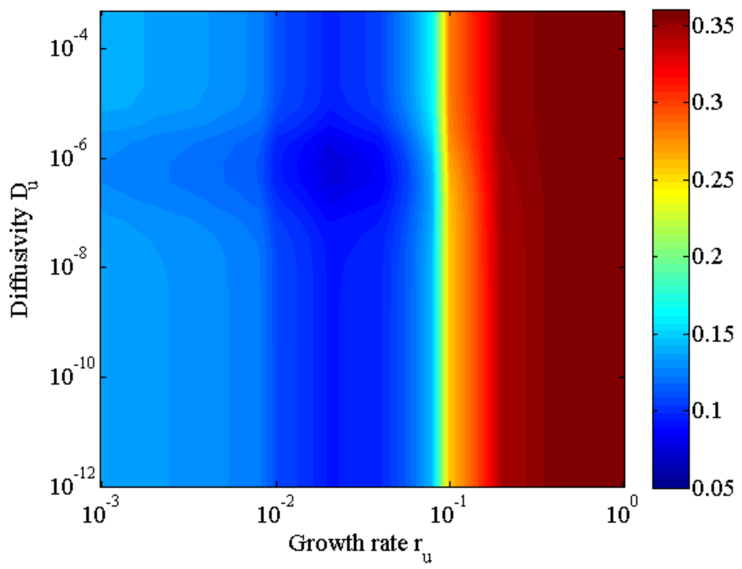


Figure 3.13: Sensitivity of error to the model parameters.

implemented and tested in models before they are implemented in practice can help enhance outcomes towards higher success rates in forest regeneration projects. Thus, the outcomes of the forest growth model can provide valuable information for land management, such as forest growth rate, the expected vegetation development status over time and the anticipation of the likely effects of forest management practices. Further, the results can be tailored for each specific location by means of accessible design parameters to be chosen according to requirements on removal efficiency (RE), mean nitrogen inflow (MNI) and vegetation cover type (VCT). In regard of watershed management, forest regeneration models can provide valuable input data for groundwater and surface-water models in what concerns the effects of vegetation on water quantity, quality and dynamics. The relationship between forest, discharge, the underground flow and the recession in a basin can be observed in a number of papers. The research carried out in Ref. [177] evaluated the discharge behavior of a water body in a forested area, thus demonstrating that the accuracy in water balance analysis is improved when this relationship is considered. Towards that, the author considers the coupling of models in which rainfall, radiation, temperature, soil moisture, canopy behavior and evapotranspiration are taken into account in order to determine the outflow of the basin in the water balance [177]. In turn, the paper in Ref. [178] presented the status of the relationship between soil occupation and water balance, plant cover and water quality [179], as well as its relation to surface and underground movement of water [180, 177]. In the studies of Ref. [181] the relationship between water loss due to interception and evapotranspiration is evaluated considering vegetation cover and its influence on the recharge process. Although the relationships between water balance, water availability, recharge, water quality, recession, movement of groundwater and surface water, level of the groundwater table to forests are accepted [182, 183, 184], the coupled modeling of forest, groundwater and surface-water seems rather unexplored from the viewpoint of physically-based coupled models. From an ecological viewpoint, the design parameters can be chosen on the basis of desired biomass production, temperature regulation, water filtration, erosion and flood control, for instance, in order to protect biotic and abiotic components of an ecosystem. We recall that, to a great extent, the restoration and regeneration of ecosystems is closely related to the regeneration of the its vegetation and forest cover.

3.4.2 Artificial neural network ensembles and the forest regeneration model as complementary tools

It has been shown recently that artificial neural network ensembles can be a responsive approach to black-box-model environmental phenomena [3]. This is particularly relevant as the amount of measured data from environmental variables steadily grows with the increasing availability of unexpensive sensors and data storage devices. Further, regarding environmental processes depending on a large number of variables, the modeling based solely on physical laws and equations becomes impracticable in most cases due to the large number of parameters involved. In this sense, the application of artificial neural networks can be of value to provide a model to those parts of the process that are not totally understood in terms of their constitutive physical, chemical or biological relations. In this context, the conceptual model (as it is the case with the diffusive-logistic forest regeneration model) can be fed with consistent inputs from the data-based black-box model. Thereby, emphasis is given to the process of interest, i.e., the process of forest regeneration.

3.4.3 Strengths and limitations of the results

The thorough application of the proposed methodology provides three main results of importance in the context of forest growth and regeneration projects: (i) prognostic maps of forest density; (ii) forest growth and diffusion parameters and (iii) performance-oriented RBS widths. A major strength of the results is that they are strongly based on data. Both the machine learning and the physical description of forest growth strongly rely on calibration, validation and test against data. Further, the ANNE allows flexible choice of input parameters type, which considerably extends the applicability of the methodology to the cases where other input variables (not necessarily those considered in this paper) are available. A strength of the implementation is it provides fast and effective tool for the solution of the model in any ordinary office computer with a CUDA GPU card. We remark that the parallel implementation achieves up to 50-fold speedup comparatively to the serial implementation. We shall provide details on performance in a separate paper. A limitation of the results that come as a result of the DLG model is that any subjacent dynamics other than diffusion and logistic growth are not captured by the model.

For instance, slower vegetation growth might be observed in a hillslope that is subjected to erosion while faster vegetation growth might be observed in a hillslope with favourable solar orientation. As a consequence, it might happen that forest density maps output by the model might contain inaccuracies as compared to real scenes, especially as such unmodelled processes become rather relevant in a given scene/time window. This issue can be properly identified by means of the in loco validation of the geoprocessing of the scene.

3.5 Final remarks

The importance of forest modeling relates to a series of factors. First, there is the need for effective forest management decision-making, which can be enhanced as the future results of current management scenarios can be projected. This seems to be particularly important as forest regeneration is a long-term process, which makes the costs and losses associated to uneffective practices and unancitipated effects potentially substantial, since they can extend for decades. Knowing the dynamics of intrinsic natural recovery is an important step to assessing the potential of resource conservation. The characterization and understanding of the typical forest regeneration process are arguably relevant in the design and evaluation of preservation projects. This research presented a self-contained framework intended to enhance and sharpen the understanding of forest regeneration and aid the design process by providing prognostics of the status of vegetation for years or decades ahead.

All the computer codes and routines developed during this research project are written in open platforms and available for download at `http://modelagemambientaluffs.blogspot.com.br/` [78].

Chapter 4

GPGPU-accelerated environmental modelling based on the 2D advection-reaction-diffusion equation[1]

T. Carlotto, R. V. da Silva, J. M. V. Grzybowski

Abstract

The advection-reaction-diffusion equation is a rather general equation that is able to describe a variety of environmental processes, such as air pollution transport, aquifer recharge, groundwater contaminant transport and forest growth. The resulting models are known to be computationally demanding, especially when the model covers large areas whereas high resolution is required. In this context, the exponential increase in computational power of massively parallel computing devices, such as the General Purpose Graphics Processing Units (GPGPUs), has paved the way to a paradigm shift in scientific computing and environmental modelling, as scientists and engineers increasingly seek to develop parallel implementations of their working models, which can take advantage of the large performance gains provided by GPGPUs. Following this trend, in this paper we present a parallel implementation of the solution of the 2D advection-reaction-diffusion equation tailored for CUDA-enabled GPGPU devices. The implementation was designed to be general and flexible in order to permit the modelling of a wide range of processes, including those featuring heterogeneous and anisotropic media. Performance tests and simulations show that the parallel implementation can outperform an equivalent sequential implementation and provides up to a 56-fold speedup. We present simulations performed with NVIDIA GTX 1060, a rather inexpensive GPU card, which show that for large grids containing over 20 million points, which corresponds to an area of 18,000 km^2 in a standard Landsat image resolution. The implementation is entirely open-source and based on free

[1]Article in development to be submitted.

platforms, thus allowing unrestricted access, use and editing. We make the commented code available for download at *modelagemambientaluffs.blogspot.com*.

4.1 Introduction

A diversity of environmental phenomena are prone to modelling by means of the so-called advection-reaction-difussion (ARD) equation. Take for example the movement of water in the soil [23, 44], the dynamics of the spatial profile and concentration of air pollution [25, 45], and the dynamics of forest growth and regeneration [24, 2]. Several biological, physical and chemical phenomena can be modelled using a combination of equations that represent specific processes that involve reaction, advection and diffusion. The advection-reaction-diffusion equation has been applied to the study of fluid dynamics [23, 44], population, forest growth and recovery [24, 2], dispersion of pollutants [25, 45], among others[46]. The mathematical models that consider advection, reaction and diffusion usually describe spatiotemporal variations in some property or variable of a process (e.g., concentration of a chemical substance, hydraulic head, vegetation density, temperature, for example).

In this context, a factor that has gained increasing attention is the parallelized solution of environmental models, which permits accurate high-resolution numerical solutions to be delivered by rather inexpensive office computers. This has become possible due to the rapid increase in the computational power of GPGPUs (General Purpose Graphics Processing Units), whose strength is to take advantage of its massively parallel architecture of memory and processors [185, 57, 142, 186, 187, 26, 188, 189].

Parallel computing consists of the simultaneous use of several processing units in order to solve a large problem by breaking it down into small parts. This requires techniques that guarantee that the combination of the solutions to each of the small parts will in the end be equivalent to the solution of the large problem posed. Besides, it is imperative in practice that the simultaneous processes observe strict control and synchronization instructions, such that the time integrity of the solution is preserved. Towards that end, process control mechanisms are necessary [190].

The ARD equation is largely applied in the modelling of environmental phenomena that involve transport along with growth or decay. The numerical solution of the advection-reaction-diffusion equation poses challenging problems, both in the field of Engineering and Computational Mathematics. A usual problem with medium to large scale simulations is that they often demand the use of supercomputers to run properly and timely. This same problem is often found also when solutions demands fine mesh grids for high resolution. The core of the concern is that usual solvers use calculation schemes based on sequential algorithms that are often too time demanding for a considerable share of practical applications. Such concern has motivated the development of alternatives to better the performance of numerical algorithms

[45, 14, 22, 188, 23]. From those, the ones based on massively parallel architectures from GPGPUs have attracted a great deal of attention due to a number of factors that make them appealing from the viewpoint of scientific computing. From those factors, one could highlight the increasing computational power and scalability of graphics processing units, along with innovating computing frameworks, such as *Compute Unified Device Architecture* (CUDA), which has made the application of graphics processing units to scientific computing accessible to scientists in general. Examples of application come from Geosciences [21, 191], Environmental Sciences [44, 11], Physics [46, 188], Mathematics [14], and many others which deal with problems demanding intensive calculation. Notably, GPGPU and the CUDA framework have permitted considerable performance gains, particularly in problems involving the solution of partial differential equations. Not only has it permitted the definition of larger computational regions, but also finer grid resolutions, both resulting in larger mesh grids [192, 46], i.e., a grid containing more points and consequently demanding more computation and memory. This technology was employed in the research by Molnár et al (2011) [45], which presents a numerical sollution of the 3D formulaton of the reaction-diffusion equation. In such study, the authors report from a 5 to a 40-fold speedup in CUDA-enabled GPGPU devices in relation to sequential CPU versions of the solution. Ji et al. 2014 [16], reported the parallelization of MODFLOW to a GPGPU framework, having reportedly obtained up to a 10-fold speedup enabling the simulation of problems involving about 10^5 cells by means of the Conjugate Gradient Method (CGM) from the CUSP and CuSPARSE library implementations [187]. Pinheiro et al. 2017 [26] presented a GPGPU parallel implementation of a model to the dispersion of pollutants in the vicinity of Brazilian nuclear power plants. The authors focused in the calculation of the wind velocity vector field on the basis of the WEST (Winds Extrapolated from Stability and Terrain) model and showed up to 25-fold speedup in relation to an equivalent sequential code processing in CPU.

In this paper, we present a GPGPU-accelerated implementation for the solution of the 2D advection-reaction-diffusion (ARD) equation. We discretize the equation using finite difference schemes and then solve the resulting linear system using Bi-Conjugate Gradient Stabilized method (BiCGStab). The results show that there is a considerable performance gain, both in terms of simulation speedup and processing boost. To propely illustrate the capabilities of the implementation, we present simulations performed with NVIDIA GTX 1060 GPU card for two case studies: (i) dispersion of particular matter PM10 from a source point, and (ii) simulation of forest recovery for an area of permanent conservation in Brazil. In order to check the upper limit of mesh grid points, further simulations are performedover large grids containing over to 20 million points, which corresponds to an area of 18,000 km² in a standard Landsat image resolution. We evaluate the performance of the implementation by applying specific performance analysis tools which revealed that the implementation kernels are processed at rates of up to 200 GFlops per second and offer substantial speedups in the overall processing time of the simulations. The outline of the paper is as follows: the materials and methods are presented in section 4.2; the case studies and performance evaluation are presented in section 4.3; the results are discussed in section 4.4; final remarks and future perspectives are given in section 4.5.

4.2 Materials and methods

This section presents the finite-difference discretization of the Advection-Reaction-Diffusion equation and the basis for the parallel implementation in CUDA.

4.2.1 The 2D advection-reaction-diffusion equation

The advection-reaction-diffusion equation (ARD) is a partial differential equation (PDE) that can be written in compact form as

$$\frac{\partial U}{\partial t}.\gamma + \nabla(-D\nabla U) + \mu\nabla(\overrightarrow{v}U) = R(U) + W(x,y,t) \tag{4.1}$$

where v_x e v_y indicate the velocity of the flow along the x and y axes, respectively, μ is mobility, W is a source (sink) term, which represents the addition (subtraction) of mass or concentration in the system, γ represents the storage capacity. The bidimensional form of the equation that assumes non-uniform velocity field and anisotropic and heterogeneous diffusion coefficients can be written as

$$\frac{\partial U}{\partial t}.\gamma - \left(D_x(U)\frac{\partial^2 U}{\partial x^2} + D_y(U)\frac{\partial^2 U}{\partial y^2}\right) + \mu\left(\frac{\partial(v_x U)}{\partial x} + \frac{\partial(v_y U)}{\partial y}\right) = R(U) + W(x,y,t) \tag{4.2}$$

where D_x and D_y are the diffusion coefficients along the the x and y axes, respectively. In the case the diffusion coefficients are assumed homogeneous and isotropic, the Eq. 4.2 simplifies to

$$\frac{\partial U}{\partial t}.\gamma - D\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) + \mu\left(\frac{\partial(v_x U)}{\partial x} + \frac{\partial(v_y U)}{\partial y}\right) = r(U) + W(x,y,t) \tag{4.3}$$

The reactive term, $r(U)$, is used to describe process of growth or decay, for example. A simple example is a process of decay in which the rate of decay is proportional to the concentration at any given time. In terms of equation, this can be written as

$$\frac{\partial U}{\partial t} = -\sigma U \tag{4.4}$$

where σ is the decay parameter, which can be a constant or a function of time or position. The decay parameter can be used to describe the process of settling of particulate matter, for example. Another reactive process of importance is the logistic growth, which is commonly used to model the dynamics of populations. Originally proposed by Verhulst [135, 136], the logistic model is represented by the differential equation

$$\frac{\partial U}{\partial t} = r_u.U\left(1-\frac{U}{k}\right) \tag{4.5}$$

where r_u is the growth rate, U is the number of individuals (which can be expressed as a proportion of the carrying capacity) and k is the carrying capacity of the environment, i.e, the number of individuals that a given environment can sustain in the long term.

4.2.2 Finite-differences discretization

The discretization of the equation in time and space was developed by assuming that the discrete points are placed in the center of each cell. The representation of the process is illustrated in Fig. 4.1. The discretization of each term of the PDE in Eq. 4.1 was performed as follows. The first-order time derivative was discretized by means of forward differences, such that

$$\frac{\partial U}{\partial t} = \frac{U^{\tau+1}-U^{\tau}}{\triangle t} \tag{4.6}$$

The term corresponding to the diffusion process is a second-order derivative in which the diffusion can possibly vary in space when heterogeneous and anisotropic media are considered. In this case, the discretization involves two separate approximations, one for external and another for internal derivatives.

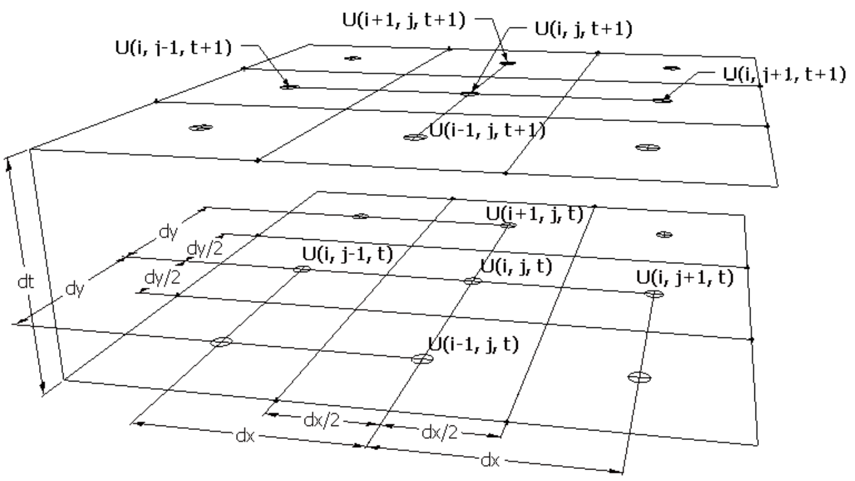


Figure 4.1: Mesh grid that illustrates the process of discretization of the ARD equation in time and space.

(i) approximation of external derivatives:

$$\frac{\partial}{\partial x}\left(D_x.\frac{\partial U}{\partial x}\right) = \frac{D_x\frac{\partial U}{\partial x}|_{x=x_{j+\frac{1}{2}}} - D_x\frac{\partial U}{\partial x}|_{x=x_{j-\frac{1}{2}}}}{\triangle x_j} \tag{4.7}$$

(ii) approximation of internal derivatives:

$$\frac{\partial}{\partial x}\left(D_x.\frac{\partial U}{\partial x}\right) = \frac{D_{x_{j+\frac{1}{2}}}\left(\frac{U_{j+1}-U_j}{\triangle x_{j+\frac{1}{2}}}\right) - D_{x_{j-\frac{1}{2}}}\left(\frac{U_j-U_{j-1}}{\triangle x_{j-\frac{1}{2}}}\right)}{\triangle x_j} \tag{4.8}$$

The term corresponding to the advection process is a first-order derivative in which the velocity fields are obtained by means of the solution of the continuity equation for the regions of interest. The application of finite-differences to the first-order term gives

$$\left[\frac{\partial(v_x U)}{\partial x}\right]_j = \frac{1}{2}\mu\left(\left[\frac{\partial(v_x U)}{\partial x}\right]_{j-\frac{1}{2}} + \left[\frac{\partial(v_x U)}{\partial x}\right]_{j+\frac{1}{2}}\right) \tag{4.9}$$

One step ahead, by adopting an algebraic notation one can write

$$\left[\frac{\partial(v_x U)}{\partial x}\right]_j = \frac{1}{2}\mu\left(\frac{v_{x_j}U_j - v_{x_{j-1}}U_{j-1}}{\triangle x_j} + \frac{v_{x_{j+1}}U_{j+1} - v_{x_j}U_j}{\triangle x_j}\right) \tag{4.10}$$

At this point, the resulting expression for the advection term agrees with that from the formulation presented in [46]. Thus:

$$\begin{aligned}\mu\left[v_x\frac{\partial U}{\partial x} + \frac{\partial v_x}{\partial x}U\right]_j &= \frac{1}{2}\mu\left(\left[v_x\frac{\partial U}{\partial x} + \frac{\partial v_x}{\partial x}U\right]_{j-\frac{1}{2}} + \left[v_x\frac{\partial U}{\partial x} + \frac{\partial v_x}{\partial x}U\right]_{j+\frac{1}{2}}\right)\\ &= \frac{1}{2}\mu\left[\left(\frac{v_{x_{j-1}}+v_{x_j}}{2}\cdot\frac{U_j-U_{j-1}}{\triangle x_j} + \frac{v_{x_j}-v_{x_{j-1}}}{\triangle x_j}\cdot\frac{U_{j-1}+U_j}{2}\right)\right] + \cdots\\ &+\frac{1}{2}\mu\left[\left(\frac{v_{x_j}+v_{x_{j+1}}}{2}\cdot\frac{U_{j+1}-U_j}{\triangle x_j} + \frac{v_{x_{j+1}}-v_{x_j}}{\triangle x_j}\cdot\frac{U_j+U_{j+1}}{2}\right)\right]\end{aligned} \tag{4.11}$$

One step ahead, the last equation can be written as

$$\left[\mu\frac{\partial(v_x U)}{\partial x}\right]_j = \frac{1}{2}\mu\left(\frac{v_{x_{j+1}}U_{j+1} - v_{x_{j-1}}U_{j-1}}{\triangle x_j}\right) \tag{4.12}$$

The formulations were developed considering the direction of the x axis and they can be analogously applied to obtain the terms corresponding to the y axis. Substituting the equations 4.6, 4.8 and 4.12 in equation 4.1 for both axis, x and y, one obtains the discretized form of the ARD equation 4.13, as

$$\frac{1}{\triangle x_j}\left[D^{\tau}_{x_{i,j+\frac{1}{2}}}\left(\frac{U^{\tau+1}_{i,j+1}-U^{\tau+1}_{i,j}}{\triangle x_{j+\frac{1}{2}}}\right)-D^{\tau}_{x_{i,j-\frac{1}{2}}}\left(\frac{U^{\tau+1}_{i,j}-U^{\tau+1}_{i,j-1}}{\triangle x_{j-\frac{1}{2}}}\right)\right]+\cdots$$

$$+\frac{1}{\triangle y_i}\left[D^{\tau}_{y_{i+\frac{1}{2},j}}\left(\frac{U^{\tau+1}_{i+1,j}-U^{\tau+1}_{i,j}}{\triangle y_{i+\frac{1}{2}}}\right)-D^{\tau}_{y_{i-\frac{1}{2},j}}\left(\frac{U^{\tau+1}_{i,j}-U^{\tau+1}_{i-1,j}}{\triangle y_{i-\frac{1}{2}}}\right)\right]+\cdots$$

$$+\left[\frac{1}{2}\mu\left(\frac{v_{x_{i,j+1}}U^{\tau+1}_{i,j+1}-v_{x_{i,j-1}}U^{\tau+1}_{i,j-1}}{\triangle x_j}\right)\right]+\cdots$$

$$+\left[\frac{1}{2}\mu\left(\frac{v_{y_{i+1,j}}U^{\tau+1}_{i+1,j}-v_{y_{i-1,j}}U^{\tau+1}_{i-1,j}}{\triangle y_i}\right)\right]+\cdots$$

$$+\sigma U^{\tau+1}_{i,j}+W_{i,j}(x,\,y,\,t)=\gamma_{i,j}\frac{U^{\tau+1}_{i,j}-U^{\tau}_{i,j}}{\triangle t} \tag{4.13}$$

In equation 4.13, it is assumed that the reaction term $r(U)$ is given by $r(U)=\sigma U^{\tau+1}_{i,j}$, which characterizes a process of decay. For any other form of reactive behavior, this term can be adequately reformulated to match the phenomenon of interst. Applying the Crank-Nicolson scheme and grouping like terms, one obtains the equation

$$\overline{E}^{\tau}_{i,j}U^{\tau+1}_{i,j}+\left[\overline{R}^{\tau}_{i,j}U^{\tau+1}_{i,j+1}+\overline{L}^{\tau}_{i,j}U^{\tau+1}_{i,j-1}+\overline{B}^{\tau}_{i,j}U^{\tau+1}_{i+1,j}+\overline{T}^{\tau}_{i,j}U^{\tau+1}_{i-1,j}+W^{\tau}_{i,j}\right]=$$

$$E^{\tau}_{i,j}U^{\tau}_{i,j}+\left[R^{\tau}_{i,j}U^{\tau}_{i,j+1}+L^{\tau}_{i,j}U^{\tau}_{i,j-1}+B^{\tau}_{i,j}U^{\tau}_{i+1,j}+T^{\tau}_{i,j}U^{\tau}_{i-1,j}-W^{\tau}_{i,j}\right] \tag{4.14}$$

in which $\overline{R}^{\tau}, \overline{L}^{\tau}, \overline{B}^{\tau}, \overline{T}^{\tau}$ are entries of the matrix $\bar{A}$ at time step τ and R^{τ}, $L^{\tau} B^{\tau} T^{\tau}$ are entries of the matrix A at time step τ. The equation 4.14 provides a generalized scheme for the solution of the ARD equation, which is a usual equation for the description of many environmental processes. The meanings and definitions of the terms in equation 4.14are presented in Table 4.1.

Table 4.1: Coefficients of equation 4.14 that composes the linear system matrices $\overline{A}$ and A.

Matrix $\overline{A}$	Matrix A
$\overline{R}_{i,j}^{\tau} = -\lambda_{i,j}\left(M_R - \mu \frac{v_{x_{i,j+1}}}{2\triangle x_j}\right)$	$R_{i,j}^{\tau} = \lambda_{i,j}\left(M_R - \mu \frac{v_{x_{i,j+1}}}{2\triangle x_j}\right)$
$\overline{L}_{i,j}^{\tau} = -\lambda_{i,j}\left(M_L + \mu \frac{v_{x_{i,j-1}}}{2\triangle x_j}\right)$	$L_{i,j}^{\tau} = \lambda_{i,j}\left(M_L + \mu \frac{v_{x_{i,j-1}}}{2\triangle x_j}\right)$
$\overline{B}_{i,j}^{\tau} = -\lambda_{i,j}\left(M_B - \mu \frac{v_{y_{i+1,j}}}{2\triangle y_i}\right)$	$B_{i,j}^{\tau} = \lambda_{i,j}\left(M_B - \mu \frac{v_{y_{i+1,j}}}{2\triangle y_i}\right)$
$\overline{T}_{i,j}^{\tau} = -\lambda_{i,j}\left(M_T + \mu \frac{v_{y_{i-1,j}}}{2\triangle y_i}\right)$	$T_{i,j}^{\tau} = \lambda_{i,j}\left(M_T + \mu \frac{v_{y_{i-1,j}}}{2\triangle y_i}\right)$
$\overline{E}_{i,j}^{\tau} = \frac{1}{\triangle t} - \left(\overline{R}_{i,j}^{\tau} + \overline{L}_{i,j}^{\tau} + \overline{B}_{i,j}^{\tau} + \overline{T}_{i,j}^{\tau} - \lambda_{i,j}\sigma\right)$	$E_{i,j}^{\tau} = \frac{1}{\triangle t} - \left(R_{i,j}^{\tau} + L_{i,j}^{\tau} + B_{i,j}^{\tau} + T_{i,j}^{\tau} + \lambda_{i,j}\sigma\right)$
$\lambda_{i,j} = \frac{1}{2\gamma_{i,j}}$	

The equations in Table 4.1 are applied to assemble the linear system of equations whose solution correspond to the numerical solution of the PDE. The terms and equations in Table 4.1 give a picture of how the processes of advection, reaction and diffusion respond to the discretization process. By adapting the coefficients from Eq. 4.1 to a particular case study, one can obtain a description of the process in terms of its advection, reaction and diffusion components. As a result, the discretized model can adapt to the description of a wide range of phenomena whose base processes are advection, reaction or diffusion. On the one hand, one can model phenomena involving advection, reaction and diffusion, or any combination of them, including those in which only one of the process is present. In order to make the discretization scheme more flexible in respect to the diffusive term, define the variables M_R, M_L, M_B, M_T, which provide the option to choose among different forms of approaching the diffusion process in relation to the characteristics of the media. While it is possible to model homogeneous and isotropic media, in which diffusivities do not change over the spatial domain, it is also possible to model heterogeneous and anisotropic media, in which diffusivities may vary in space and even assume different values along the x and y axes. In the latter case, it becomes necessary to recurr to the calculation of the mean value of diffusivities for neighboring cells in order to avoid discontinuities in the diffusivity parameter among cells.

The definition of the variables M_R, M_L, M_B, M_T are presented in Table 4.2. The table also brings the expressions for the calculation of mean values of diffusivities according to three different methodologies: (i) harmonic mean, (ii) arithmetic mean, (iii) geometric mean.

By assuming different combination of the terms in Tables 4.1 and 4.2, the resulting discretized model in Eq.4.14 can be adjusted in order to describe the phenomena in homogeneous

Table 4.2: Formulations of the diffusive term for homogeneous and heterogeneous media. For heterogeneous media, the value of the parameters for a given point is calculated by means of arithmetic, harmonic or geometric mean.

	Characteristics of the media		
Component of the diffusion term	Heterogeneous and anisotropic		
	Arithmetic mean	Harmonic mean	Geometrical mean
M_R	$\frac{\left[\frac{D^{\tau}_{x_{i,j}}\triangle x_{j+1}+D^{\tau}_{x_{i,j+1}}\triangle x_j}{2(\triangle x_{j+1}\triangle x_j)}\right]}{\triangle x_j}$	$\frac{\left[\frac{2D^{\tau}_{x_{i,j}}D^{\tau}_{x_{i,j+1}}}{D^{\tau}_{x_{i,j}}\triangle x_{j+1}+D^{\tau}_{x_{i,j+1}}\triangle x_j}\right]}{\triangle x_j}$	$\frac{\sqrt{\frac{D_{x_{i,j}}D_{x_{i,j+1}}}{\triangle x_i \triangle x_{i+1}}}}{\triangle x_j}$
M_L	$\frac{\left[\frac{D^{\tau}_{x_{i,j}}\triangle x_{j-1}+D^{\tau}_{x_{i,j-1}}\triangle x_j}{2(\triangle x_{j-1}\triangle x_j)}\right]}{\triangle x_j}$	$\frac{\left[\frac{2D^{\tau}_{x_{i,j}}D^{\tau}_{x_{i,j-1}}}{D^{\tau}_{x_{i,j}}\triangle x_{j-1}+D^{\tau}_{x_{i,j-1}}\triangle x_j}\right]}{\triangle x_j}$	$\frac{\sqrt{\frac{D^{\tau}_{x_{i,j}}D^{\tau}_{x_{i,j-1}}}{\triangle x_i \triangle x_{i-1}}}}{\triangle x_j}$
M_B	$\frac{\left[\frac{D^{\tau}_{y_{i,j}}\triangle y_{i+1}+D^{\tau}_{y_{i+1,j}}\triangle y_i}{2(\triangle y_{i+1}\triangle y_i)}\right]}{\triangle y_i}$	$\frac{\left[\frac{2D^{\tau}_{y_{i,j}}D^{\tau}_{y_{i+1,j}}}{D^{\tau}_{y_{i,j}}\triangle y_{i+1}+D^{\tau}_{y_{i+1,j}}\triangle y_i}\right]}{\triangle y_i}$	$\frac{\sqrt{\frac{D^{\tau}_{y_{i,j}}D^{\tau}_{y_{i,j+1}}}{\triangle y_i \triangle y_{i+1}}}}{\triangle y_j}$
M_T	$\frac{\left[\frac{D^{\tau}_{y_{i,j}}\triangle y_{i-1}+D^{\tau}_{y_{i-1,j}}\triangle y_i}{2(\triangle y_{i-1}\triangle y_i)}\right]}{\triangle y_i}$	$\frac{\left[\frac{2D^{\tau}_{y_{i,j}}D^{\tau}_{y_{i-1,j}}}{D^{\tau}_{y_{i,j}}\triangle y_{i-1}+D^{\tau}_{y_{i-1,j}}\triangle y_i}\right]}{\triangle y_i}$	$\frac{\sqrt{\frac{D^{\tau}_{y_{i,j}}D^{\tau}_{y_{i,j-1}}}{\triangle y_i \triangle y_{i-1}}}}{\triangle y_j}$
For homogeneous media, the equations reduce to: $M_R = M_L = \frac{D_x}{\triangle x^2}$ and $M_B = M_T = \frac{D_y}{\triangle y^2}$.			

and isotropic or heterogeneous and anisotropic media. It also permits the treatment of diffusion-only, advection-only or reaction-only processes, or any combination of them. For instance, as one combines advection and diffusion along with the velocity field, one obtains a formulation which is adequate for dispersion.

From Eq. 4.14 and considering the expressions presented in Tables 4.1 and 4.2, the linear system is given as

$$\overline{A}^{\tau}U^{\tau+1} = A^{\tau}U^{\tau} + W^{\tau} \tag{4.15}$$

$$U^{\tau+1} = \left(\overline{A}^{\tau}\right)^{-1} A^{\tau}U^{\tau} + W^{\tau} \tag{4.16}$$

We solve Eq. 4.16 with the implementation of the Bi-Conjugate Gradient Stabilized (BiCGStab) algorithm from the CUSP library is chosen. The library contains the parallel implementation of the BiCGStab algorithm that are tailored for GPGPU.

4.2.3 GPGPU-accelerated numerical solution of the 2D advection-reaction-diffusion equation

The implementation of the model was developed using the *NVIDIA*® CUDA language by means of the CUDA ToolKit v8.0. The hardware GPGPU used for the simulations was a GeForce GTX 1060 graphic card running in an *Intel*® *core*™ $i7-7700K$ PC. A usual feature of computing using CUDA is the combined application of CPU and GPU resources. While tasks that demand little computational power are usually performed in the CPU, the tasks that demand massive calculations are performed in the GPU. The flowchart for the algorithm is presented in Fig. 4.2.

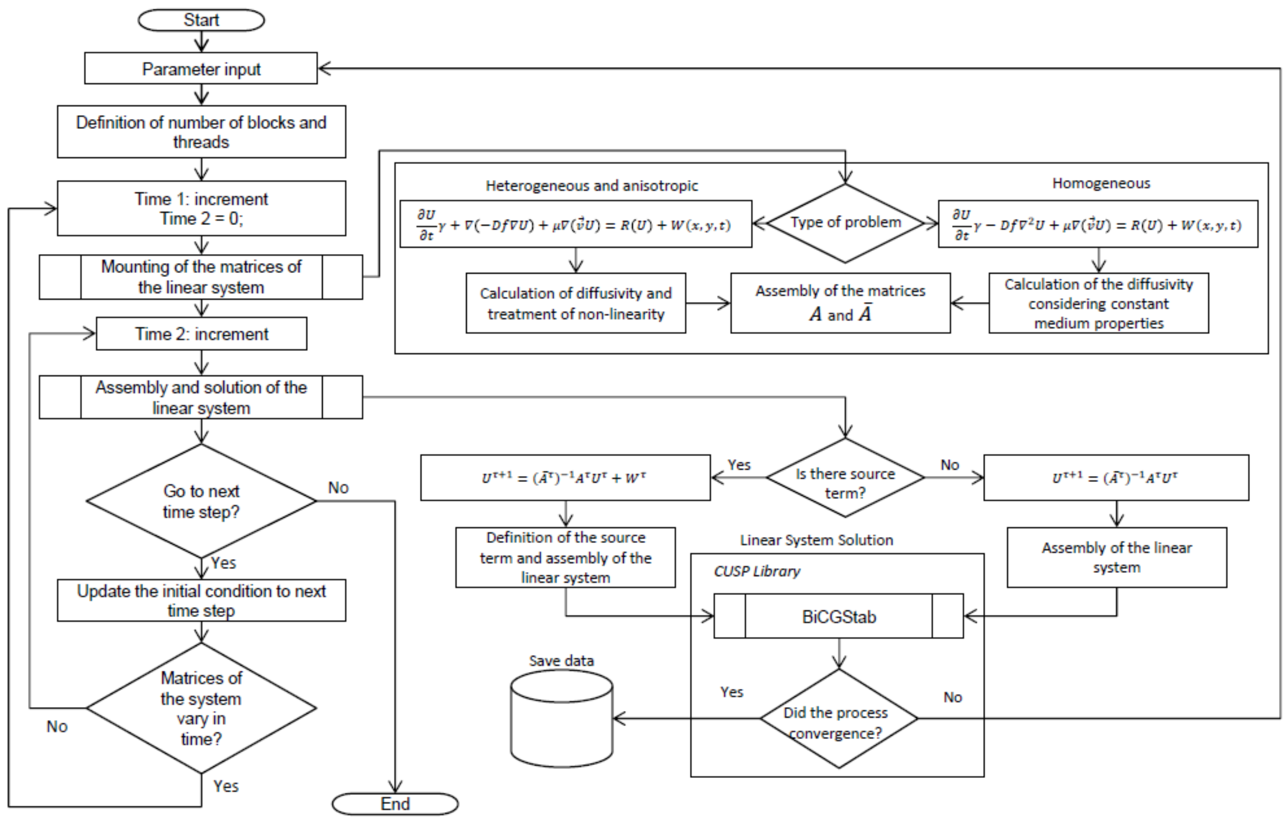
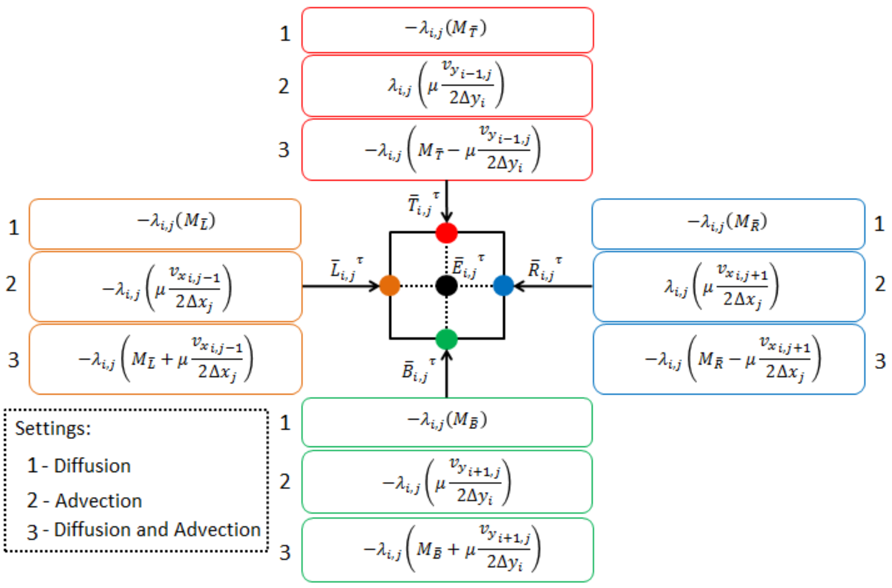

Start
Parameter input
Definition of number of blocks and threads
Time 1: increment
Time 2 = 0;
Mounting of the matrices of the linear system
Time 2: increment
Assembly and solution of the linear system
Go to next time step?
No
Yes
Update the initial condition to next time step
Matrices of the system vary in time?
No
Yes
End
Heterogeneous and anisotropic
$\frac{\partial U}{\partial t}\gamma + \nabla(-Df\nabla U) + \mu\nabla(\vec{v}U) = R(U) + W(x,y,t)$
Type of problem
Homogeneous
$\frac{\partial U}{\partial t}\gamma - Df\nabla^2 U + \mu\nabla(\vec{v}U) = R(U) + W(x,y,t)$
Calculation of diffusivity and treatment of non-linearity
Assembly of the matrices A and $\bar{A}$
Calculation of the diffusivity considering constant medium properties
$U^{\tau+1} = (\bar{A}^{\tau})^{-1}A^{\tau}U^{\tau} + W^{\tau}$
Yes
Is there source term?
No
$U^{\tau+1} = (\bar{A}^{\tau})^{-1}A^{\tau}U^{\tau}$
Definition of the source term and assembly of the linear system
Linear System Solution
CUSP Library
Assembly of the linear system
BiCGStab
Save data
Yes
Did the process convergence?
No

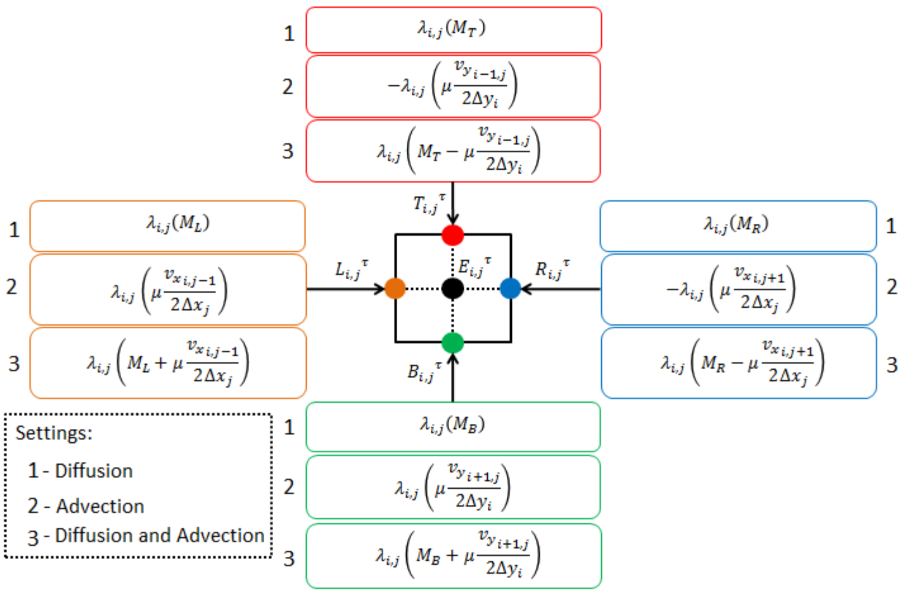
Figure 4.2: Flowchart of the GPGPU-accelerated implementation of the 2D advection-reaction-diffusion equation.

One of the most important steps in the algorithm is the mounting of the matrices A and $\bar{A}$. The mounting of the matrices takes into consideration the processes included in the description of the phenomenon and their parameters. The numbering (1,2) as shown in Fig. 4.3 and 4.4 indicate the discretized terms for the processes of diffusion (1) and advection (2). The combination of both processes gives origin to the process of dispersion. In short, the terms that are included in the mounting of the matrices A and $\bar{A}$ for a given model indicate the processes that are involved in the phenomenon described by that model.



$-\lambda_{i,j}(M_{\bar{T}})$
$\lambda_{i,j}\left(\mu\frac{v_{y\,i-1,j}}{2\Delta y_i}\right)$
$-\lambda_{i,j}\left(M_{\bar{T}}-\mu\frac{v_{y\,i-1,j}}{2\Delta y_i}\right)$
$-\lambda_{i,j}(M_{\bar{L}})$
$-\lambda_{i,j}\left(\mu\frac{v_{x\,i,j-1}}{2\Delta x_j}\right)$
$-\lambda_{i,j}\left(M_{\bar{L}}+\mu\frac{v_{x\,i,j-1}}{2\Delta x_j}\right)$
$-\lambda_{i,j}(M_{\bar{R}})$
$\lambda_{i,j}\left(\mu\frac{v_{x\,i,j+1}}{2\Delta x_j}\right)$
$-\lambda_{i,j}\left(M_{\bar{R}}-\mu\frac{v_{x\,i,j+1}}{2\Delta x_j}\right)$
$-\lambda_{i,j}(M_{\bar{B}})$
$-\lambda_{i,j}\left(\mu\frac{v_{y\,i+1,j}}{2\Delta y_i}\right)$
$-\lambda_{i,j}\left(M_{\bar{B}}+\mu\frac{v_{y\,i+1,j}}{2\Delta y_i}\right)$
$\bar{T}_{i,j}{}^{\tau}$ $\bar{L}_{i,j}{}^{\tau}$ $\bar{E}_{i,j}{}^{\tau}$ $\bar{R}_{i,j}{}^{\tau}$ $\bar{B}_{i,j}{}^{\tau}$
Settings:
1 - Diffusion
2 - Advection
3 - Diffusion and Advection

Figure 4.3: Scheme of the operations performed during the mounting of the matrix $\bar{A}$ of the linear system in Eq. 4.16.



$\lambda_{i,j}(M_T)$
$-\lambda_{i,j}\left(\mu\frac{v_{y\,i-1,j}}{2\Delta y_i}\right)$
$\lambda_{i,j}\left(M_T-\mu\frac{v_{y\,i-1,j}}{2\Delta y_i}\right)$
$\lambda_{i,j}(M_L)$
$\lambda_{i,j}\left(\mu\frac{v_{x\,i,j-1}}{2\Delta x_j}\right)$
$\lambda_{i,j}\left(M_L+\mu\frac{v_{x\,i,j-1}}{2\Delta x_j}\right)$
$\lambda_{i,j}(M_R)$
$-\lambda_{i,j}\left(\mu\frac{v_{x\,i,j+1}}{2\Delta x_j}\right)$
$\lambda_{i,j}\left(M_R-\mu\frac{v_{x\,i,j+1}}{2\Delta x_j}\right)$
$\lambda_{i,j}(M_B)$
$\lambda_{i,j}\left(\mu\frac{v_{y\,i+1,j}}{2\Delta y_i}\right)$
$\lambda_{i,j}\left(M_B+\mu\frac{v_{y\,i+1,j}}{2\Delta y_i}\right)$
$T_{i,j}{}^{\tau}$ $L_{i,j}{}^{\tau}$ $E_{i,j}{}^{\tau}$ $R_{i,j}{}^{\tau}$ $B_{i,j}{}^{\tau}$
Settings:
1 - Diffusion
2 - Advection
3 - Diffusion and Advection

Figure 4.4: Scheme of the operations performed during the mounting of the matrix A of the linear system in Eq. 4.16.

Regarding the assignment of the boundary conditions, the cells belonging to the borders of the computational domain were specified to have two types of boundaries: Neumann, in which it is assigned a condition of non-flux or Dirichlet, in which the boundary is assigned a fixed value. These boundary conditions suffice to describe most usual situations. The scheme corresponding to the Neumann bondary condition are presented in Fig. 4.5.

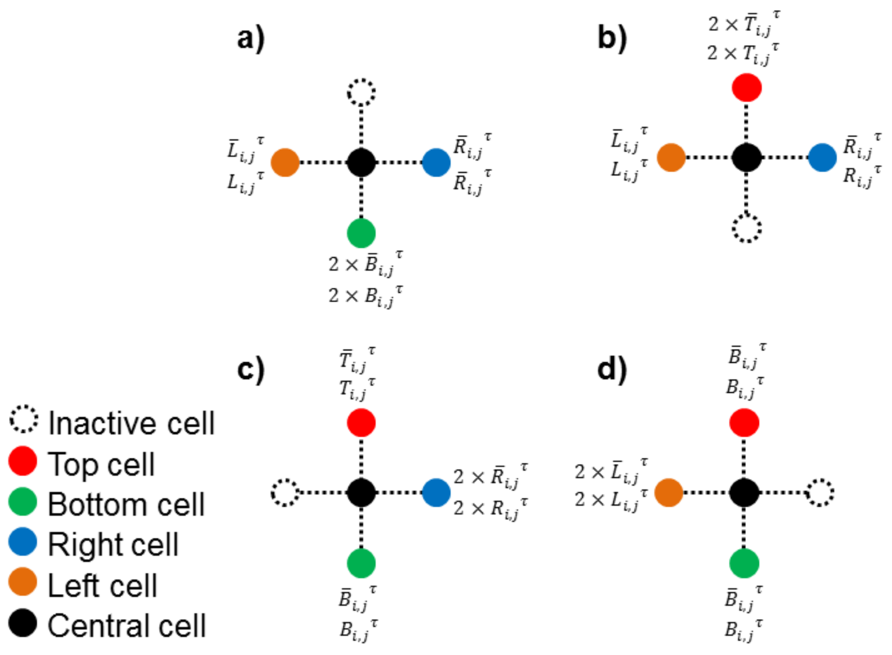


Figure 4.5: Treatment of Neumann boundary conditions for the condition of zero flow applied at the cells in the vicinity of the edges of the computational domain.

The definition of the computational domain, i.e., the active cells in the mesh grid, was set to be processed in the CPU by means of a sequential code. The code maps and stores the active cells in the computational domain, i.e., the cells that will be used during the calculation of the solution. Active and inactive cells are identified by means of an auxiliary matrix. Both regular rectangular and irregular grids of active cells are admitted, such that the program permits the solution over arbitrary 2D grids. The auxiliary matrix is also used to define the type of boundary condition for boundary nodes. An example of application of the auxiliary matrix is presented in Fig. 4.6. The active nodes are numbered sequentially in a location matrix, as shown in Fig. 4.6. The entries of the location matrix are used to check if an active node neighbors a boundary and to map the vicinity of each node in terms of neighboring nodes or boundaries. By combining the information acquired in the auxiliary matrix and in the location matrix, one can mount the coefficient matrices of the linear system, A and $\bar{A}$.

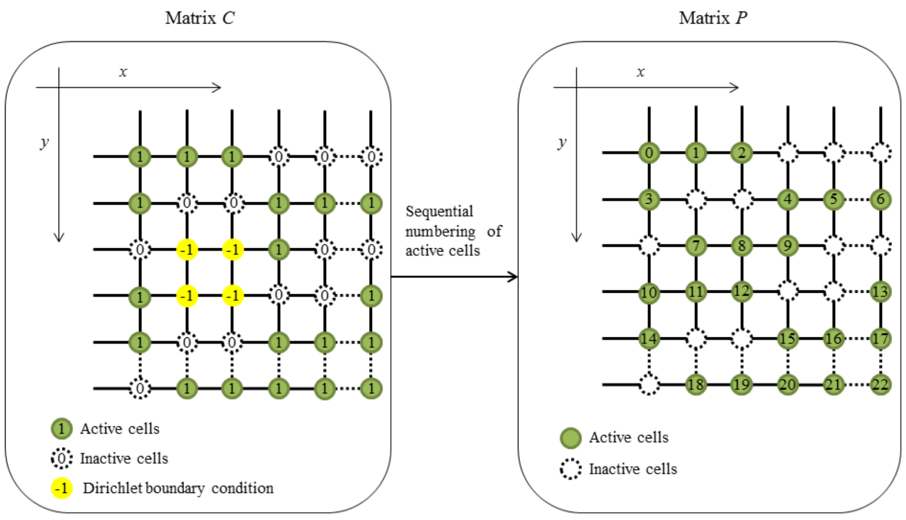

Matrix C
Matrix P
x
y
Sequential numbering of active cells
Active cells
Inactive cells
Dirichlet boundary condition

Figure 4.6: Definition of the matrices C (contours) that stores the location and the type of boundary conditions and P (positions) that define the locations of the active cells in the computational domain.

The computational domain is the set of active cells in the meshgrid, i.e., the set of cells where the calculations are performed. The computational domain can comprise the entire meshgrid or a subset of its cells. Thus, during the process of mounting the matrices A and $\bar{A}$, the cells that do not belong to the computational domain are disregarded and the operations are oriented according to the entries of the location matrix. From the location matrix or from the auxiliary matrix the dimension of the linear system to be solved can be defined. Although the illustrations in Fig. 4.6 show the auxiliary and location matrices represented as bidimensional arrays, in order to make them more explanatory of the process of mounting of the linear system, such matrices are more actually stored in the form of vectors for performance and reduced storage requirements.

Assignment of the number of blocks and threads

Once the size of the linear system is determined, the computing workload required for its solution has to be shared among the available streaming multiprocessors and single processors in the GPGPU. The way this is accomplished is by dividing the processing in blocks containing a number of threads that will favour and optimize the use of the hardware. The number of threads is defined by the number of single processors available in a streaming multiprocessor unit. In our implementation, the code compares the number of threads (say, n threads) available per block and divides the workload in N blocks with size n, i.e., n threads. In this case, N represents the integer part of the division of the linear system size by the number of available threads.

Mounting of the linear system matrix in GPGPU

The mounting of the matrices A and $\bar{A}$ is the stage at which the position of the active points in the mesh and the dynamics to which they are subject are both stored in the form of coefficient matrices to a linear system. The proper mounting of the matrices preserves the physical attributes of the system and that makes it a very important step towards the solution. The first step to the define the procedure to be followed in the mounting of the matrices is to check what the characteristics of the media are. If the media is homogeneous and isotropic, the mounting of the matrices is based on Eq. 4.3 and the problem is simplified since the calculation of the mean is unnecessary. If the media is heterogeneous, the mounting is based on Eq. 4.2 and it requires the calculations of the means to smooth the transition between regions of the media with different parameter values.

In addition, if the parameters depend on the value of U, as it is the case of the soil transmissivities, which increases with hydraulic head, the entries of the matrix must be updated at every time step. Besides, in this case the system becomes nonlinear and the mounting process repeats right after the values of U in the mesh are available. In this case, the computational burden is considerably higher, such that this processed is assigned for calculation at the GPGPU by means of a kernel designed to this end. The matrices are stored in CRS (Compressed Row Storage) format, which is adequate for the storage of sparse matrices.

Solution of the linear system

The linear system resulting from the discretization of the equations, in Eq. 4.16, is solved by means of the Bi-Conjugate Gradient Stabilized (BiCGStab) method. The implementation of BiCGStab from the Cusp library is applied for the solution in the GPGPU.

Storage of the results

The solution of the equations are stored in the formats matrix market (.mtx), text (.txt) and keyhole markup language (.kml). The matrix market outputs contain the solution of the system for each time step and they can be processed in softwares of data visualization, such as *VisIt* [193]. The output text files can be processed in GIS softwares, such as GRASS GIS [143]. The keyhole markup language files can be visualized in Google Earth or Google Maps, for example, both in computers or mobile phones [194].

4.2.4 Performance evaluation of the implementation

The performance takes into account the elapsed time for the processing of the simulation and the number of Gigaflops (GFLOPs) per second as a function of the number of grid points for each GPGPU kernel. Such information was obtained by means of the tool *NVIDIA®Nsight*™

Visual Studio Edition 5.4, which is an development environment tailored for CUDA C/C++ applications. The resources from *NVIDIA®Nsight™* permit the collection of the statistics of applications developed in CUDA C/C++. Such statistics allows developers and users to evaluate how the computational burden is distributed among the processors, to follow the memory usage and the execution of the kernels and to monitor the calling and execution of CPU and GPU tasks.

In this research, such tools were used to generate performance reports that allow the evaluation of the processing time of kernels for the two types of solution implemented in the code: (i) the solution considering constant parameters and (ii) the solution considering time-varying parameters, in which the linear system matrices have to be reconstructed at each time step. The analysis consisted of determining the duration of the processing of each kernel and how they contribute to the total elapsed time until the solution is delivered. To relate the performance evaluation with the size of the problem under study, we monitor the performance of the models for grids with an increasing number of nodes until the processing capability of the NVIDIA GTX 1060 GPGPU card is exceeded.

4.3 Results

This section illustrates the application of the GPGPU-accelerated implementation of the numerical solution for the 2D advection-reaction-diffusion equation. To this end, two case studies are presented: (i) numerical simulation of the dispersion of coarse particulate matter (PM10) from a quarry, as part of an environmental impact study of the proposed development of an open-pit site for rock extraction; (ii) numerical simulation of forestry recovery at State Biological Reserve of Mata Paludosa, a conservation unit in the state of Rio Grande do Sul, Brazil, in which transitional areas between hillside and lowland environment are permanently protected since the year 1998.

To properly illustrate the application and capabilities of the GPU-accelerated implementation, we ommit excessive technical and operational details (that can be found elsewhere [195, 13]) and focus on the production and visualization of the solution to the environmental model. All the simulations were performed in an NVIDIA GTX 1060 GPU card and Intel $i7-7700$ CPU.

4.3.1 Case study 1: Assessing the spatial profile of PM_{10} concentration in the vicinity of a quarry

Quarrying is widely acknowledged to contribute to particulate matter emissions [196]. For this reason, environmental impact studies preceding the installation of quarries are important to foresee the likely effects of such activity in particulate matter air pollution. In this case study,

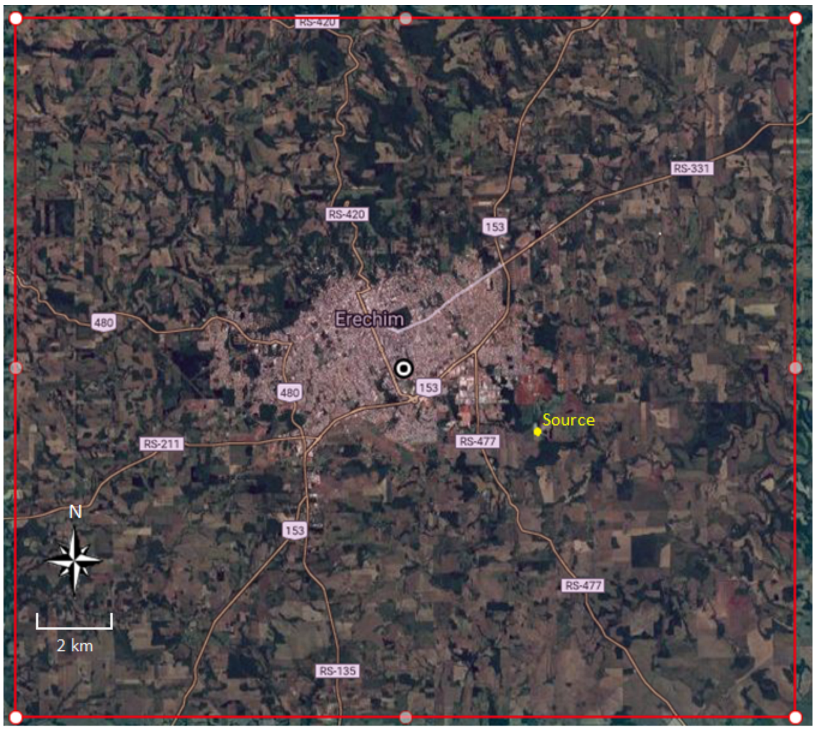


Figure 4.7: Application area for case study of PM_{10} dispersion. The red polygon delimits the computational area and the yellow dot shows the source point of pollution.

we present the simulation of dispersion of particulate matter from an active quarry for particle sizes smaller than $10\mu m$, the so-called PM_{10}. The exposition to particulate matter air pollution is reported to have harmful effects upon human health [197]. Coarse mode particles, or PM_{10} originate from resuspended dust, soil dust, road dust, metal oxides and is generated by farming, mining, construction and the combustion of oil and coal [197]. Coarse mode particles typically travel from 1 to 10 kilometers and settle in minutes or hours. The area under study is limited by the coordinates 27° 33’ 47”S, 27° 43’ 47”S, 52° 09’ 32”W, 52° 22’ 8”W and it contains a point source of particulate matter that during operation produces a steady concentration of $1000\mu g.m^{-3}$[198] and is located at the coordinates 27° 39’ 39” S and 52° 13’ 43 W.

The mean wind speed at 10 meters height to feed the advective term of the equation was obtained from the Global Forecast System (GFS) by means of the WindNinja software [7]. The digital elevation model was obtained from the SRTM 90m Digital Elevation Database [199]. The characteristic diffusion parameter for quarry and open-pit mining activities was obtained from the literature [200], as well as the concentration of particulate matter [200]. The horizontal diffusion coefficient was set to $10^4 m^2.s^{-1}$ [201]. The results from the simulation performed in WindNinja for mean wind speeds over the computational region is presented in Fig. 4.8.

The simulation of dispersion of PM_{10} is presented in Fig. 4.9. The total time of simulation for a mesh grid with 97,236 points and 1,200 integration steps was about 3,000 seconds in the parallel GPGPU implementation. For improved visualization of the results, the solution of the

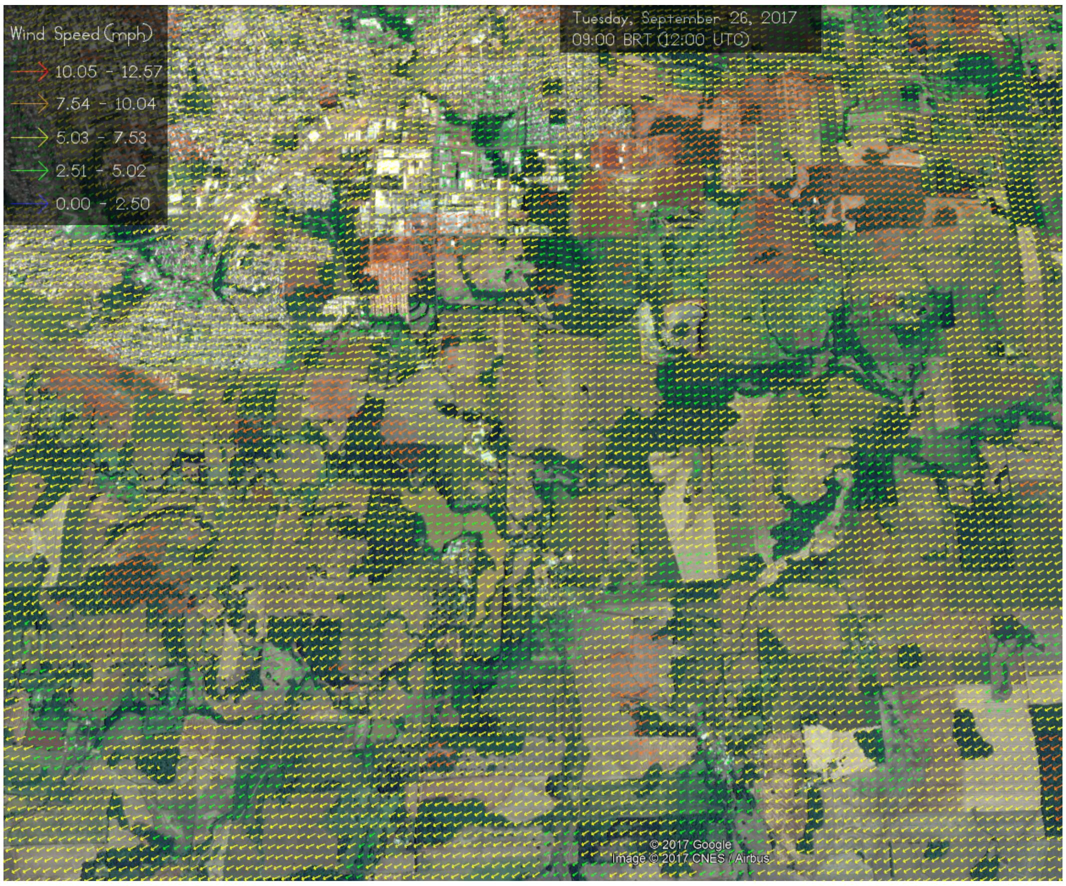


Figure 4.8: Velocity field for the computational area, obtained from the Global Forecast System (GFS) by means of the software WindNinja [7].

equation was exported to KML format and is processed in the software Google Earth. Further information on the performance of the implementation is presented in the subsection 4.3.3.

4.3.2 Case study 2: Assessing forest recovery for forestry management and planning

Forest recovery is widely acknowledged to be among the most important activities to the preservation of habitats, ecosystems, water bodies and water quality in catchment basins [121, 103, 111, 3]. The application of mathematical models, such as the diffusive-logistic growth (DLG) model was shown to provide helpful information regarding the mid and long-term behavior of preservation areas undergoing a recovery process [24, 2]. The DLG model is a particular case of the ARD equation for which the advection term vanishes and, as such, is feasible of study using our parallel implementation of the ARD equation.

In this case study, we consider the forest recovery process occurring at the State Biological Reserve of Mata Paludosa, which is under permanent conservation since it became a conservation area by decree in 1998. It features transitional areas between hillside and lowland environments which shelter remnants of forest formed on very humid soils, interspersed with the

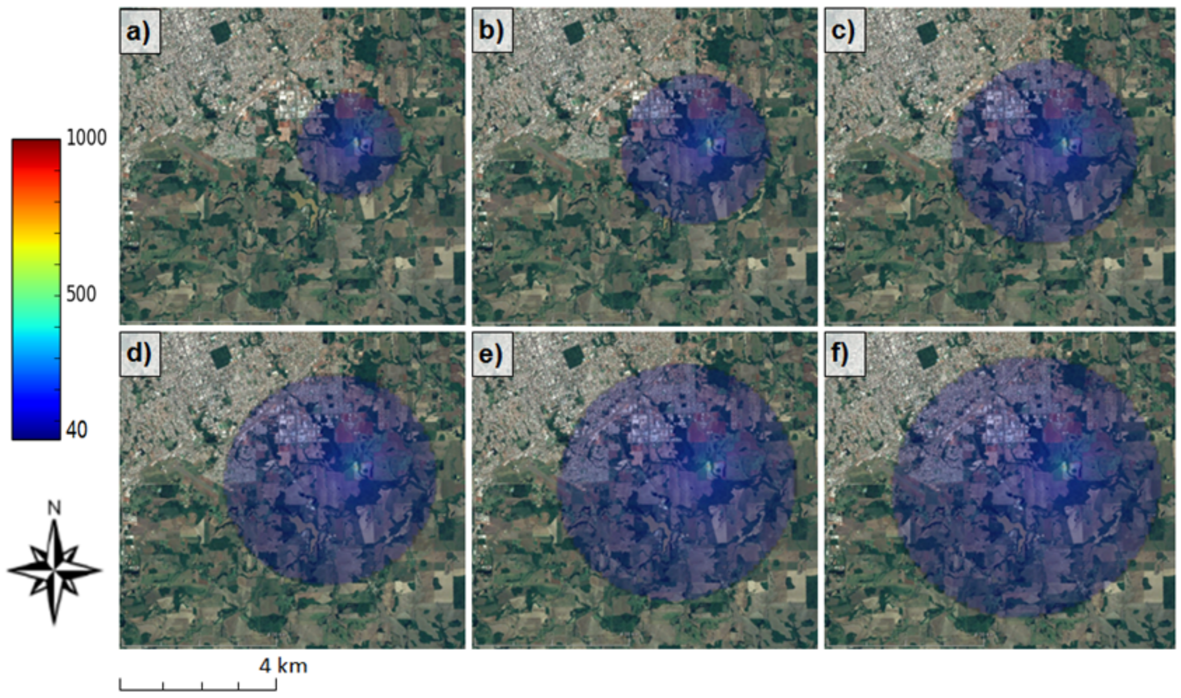


Figure 4.9: Simulation of dispersion of particulate matter PM_{10} from a quarry: the sequence of figures show the evolution of the plume for a) 3 minutes, b) 6 minutes, c) 10 minutes, d) 13 minutes, e) 16 minutes and f) 20 minutes.

vegetation of plains. The typical vegetation of this formation includes epiphytes, bromeliads, orchids and palm trees, especially palmetto-juçara (Euterpe edulis), gamiova (Genoma gamiova) and Guarani (Genoma Schottiana), which are in danger of extinction. This formation plays an important role in the conservation of some amphibians and birds which only occurr in this type of environment.

The calibration of the model followed the procedures described in Ref. [2]. The area considered in the case study is located between longitudes 29°29’58”S and 29º31’20”S and between latitudes 50º05’28”W and 50º07’59”W. The region is shown in Fig. 4.10.

The Landsat scenes usef for calibration, validation a application of the model were acquired from the Landsat 5 TM and Landsat 8 TM database from the US Geological Survey (USGS). The Landsat 5 TM images are composed of 7 spectral bands (Blue, Green, Red, Near-IR, Mid-IR, Thermal-IR, SW-IR, Panchromatic), while the Landsat 8 TM are composed of 12 spectral bands (Coastal, Aerosol, Blue, Green, Red, Near-IR, SWIR1, SWIR2, Panchromatic, Cirrus, TIR1, TIR2). The process of obtaining the vegetation density for calibration, validation and application of the model is explained in detais in Ref. [2]. In summary, the vegetation index is based on the calculation of the Enhanced Vegetation Index (EVI), by means of the equation

$$EVI = G \cdot \frac{\rho_{NIR} - \rho_{red}}{\rho_{NIR} + C_1 \cdot \rho_{red} - C_2 \cdot \rho_{blue} + L} \tag{4.17}$$

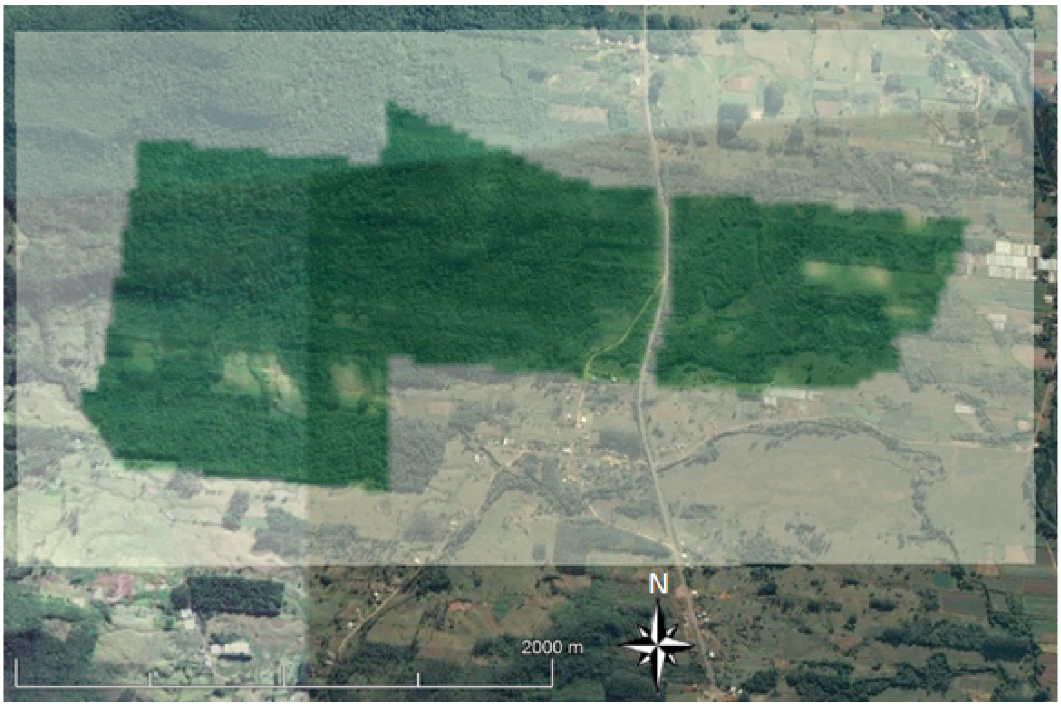


Figure 4.10: Application area for the model of forest growth. The illustration highlights a portion of forest from the State Biological Reserve of Mata Paludosa.

where ρ_{red},ρ_{blue} and ρ_{NIR} represent the red, blue and near infrared (NIR) spectral bands, G is a gain factor, $C_1$and C_2 are ajustable coefficients that are related to the presence of aerosols, L is a parameter which takes into account effects of the soil on reflectance. The standard values for these parameters, as commonly adopted in the literature, are $L = 1$, $C_1 = 6$, $C_2 = 7,5$ and $G = 2,5$ [202, 203]. The EVI is considered to give an accurate account of vegetation density in biomes with dense forest and agricultural areas, while this set of parameters are regarded as fairly robust to applications with Landsat TM images [202]. The images used for calibration and validation of the model are from the years 2005 and 2015.

The simulation in Fig. 4.11 presents the solution of the DLG model for a period of 10 years comprising the period from March, 2005 to March, 2015. The total time of simulation for a mesh grid with $3,028$ points and 10 integration steps was 1.5 second in the parallel GPGPU implementation. For improved visualization of the results, the solution of the equation was exported to KML format and is processed in the software Google Earth. Further information on the performance of the implementation is presented in the subsection 4.3.3.

4.3.3 Time performance of the GPGPU-accelerated implementation

The performance tests were performed individually for each kernel processed by the GPGPU. Table 4.3 shows the number of blocks and threads for each mesh grid size and the computation time for the diffusive-logistic growth model. In this process, the linear system is mounted once

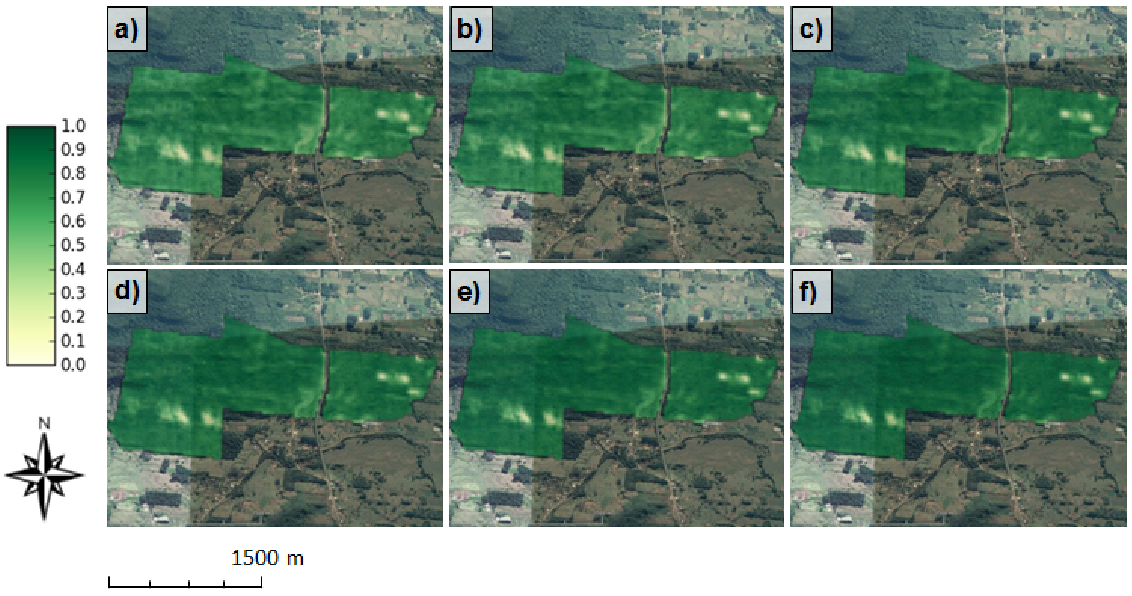


Figure 4.11: Simulation of forest growth for a period of 10 years in the State Biological Reserve of Mata Paludosa: the simulation results were saved in KML format for visualization in the software Google Earth. The simulation frames correspond to forest density at (a) the initial condition and after (b) 2 years, (c) 4 years, (d) 6 years, (e) 8 years and (f) 10 years. Vegetation density is normalized to the interval $[0,1]$.

and the system is repeatedly solved for each iteration. For this reason, most of the simulation time is spent in the solution of the linear system, which is calculated for each time step over the iterations.4.13.

Table 4.3: Elapsed time for the execution of the main kernels used in the forest regeneration model as a function of the number of mesh grid points.

			Elapsed time (μs)	
Grid points	Blocks	Threads	par_const_TxTy	ClassMatPOS
2,851,200	2,785	1,024	318.496	2,992.160
11,404,800	11,138	1,024	1,226.816	9,739.072
13,799,800	13,477	1,024	1,479.200	11,743.168
16,422,900	16,038	1,024	1,765.536	13,955.232
19,274,100	18,823	1,024	2,069.760	16,314.752
20,785,200	20,299	1,024	2,253.472	17,545.600

One way of measuring the performance of GPU kernels is to determine the number of floating point operations per second (FLOP/s) that are performed during the solution procedure. The evaluation of performance is presented for the two main kernels of the GPGPU implementation of the 2D advection-reaction-diffusion equation. The number of FLOP/s as a function of the number of points in the computational mesh grid are presented in Table 4.3. From the data, it can be observed that the kernels operate perform from 80 to 200 billion floating point operations per second to computational mesh grids varying from 2 to 20 million cells (points). To evaluate the performance of the implementation for a process that demands the reconstruction of the linear system at every integration step, we consider next the case of groundwater modelling of unconfined aquifers. In this case, the transmissivities are usually modelled as a function of hydraulic head. The simulation data are shown in Table 4.4.

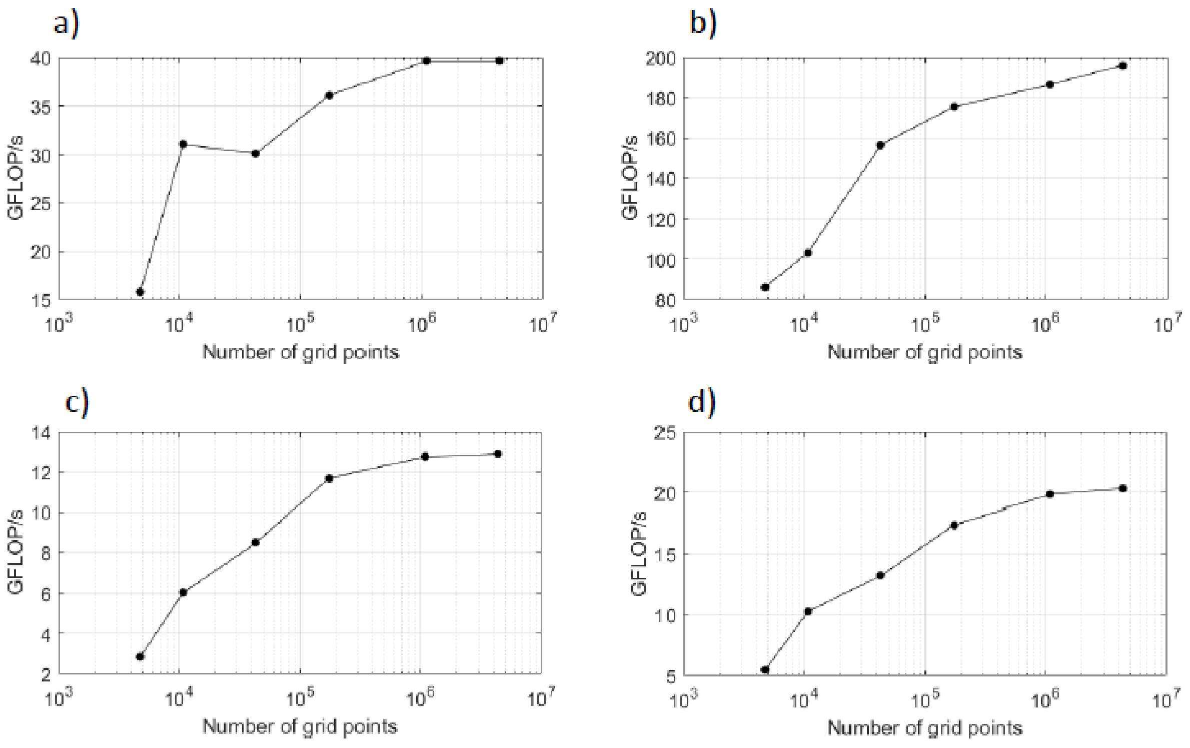


Figure 4.12: Performance in FLOPs per second for each kernel of the model describing groundwater flow: a) kernel par_var_TxTy, which calculates the time-varying parameters for the mouting of the linear system, b) kernel ClassMatPOS, which computes the entries of the matrices of the linear system, $\bar{A}$ and A, c) kernel subtract_vet, which performs vector subtractions for every other routines and d) kernel gw_baseflow, which solves the groundwater flow problem.

Table 4.4: Elapsed time for the execution of the main kernels used in the groundwater flow model as a function of the number of mesh grid points.

			Elapsed time (μs)			
Grid points	Blocks	Threads	par_var_TxTy	ClassMatPOS	subtract_vet	gw_baseflow
4,762	2,785	1,024	3.904	17.696	2.112	4.320
10,767	11,138	1,024	4.512	33.600	2.304	5.216
43,347	13,477	1,024	18.720	89.920	6.688	16.320
173,538	16,038	1,024	62.496	322.848	19.616	49.792
1,085,798	18,823	1,024	355.968	1,904.448	113.184	273.568
4,342,908	20,299	1,024	1,424.064	7,254.336	448.512	1,066.752

The Fig. 4.12 present the performance in FLOPs per second for each kernel of the model

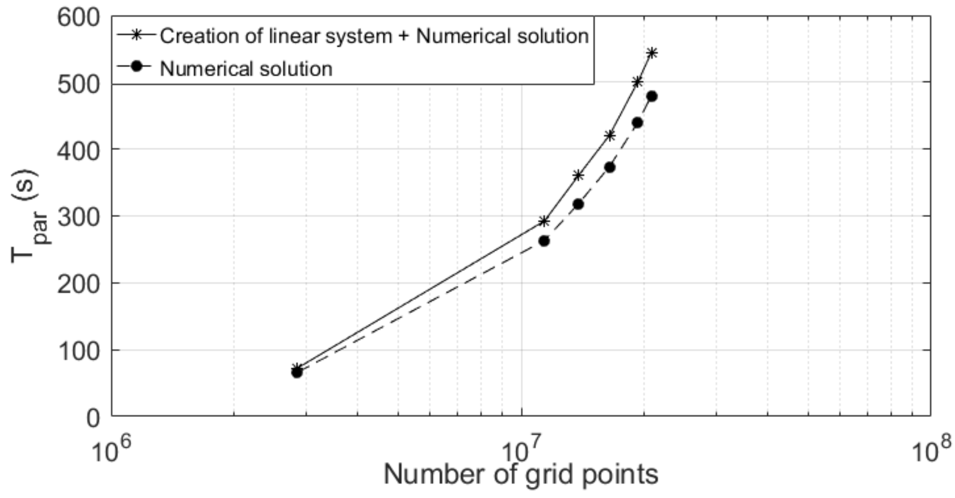


Figure 4.13: Elapsed time until the solution of the model as a function of the number of grid points in the mesh.

describing groundwater flow. The kernel ClassMatPOS, which is responsible for the mounting of the linear system features a larger number of FLOP/s in relation to the remaining kernels, of the order of 200 billion operations per second. Although the performance of the kernels can be considered satisfactory in respect to the total time of simulation, the number of FLOPs per second observed during the execution of the routine lies well below the peak performance of the GPU GTX 1060 6Gb card, which is reported to be 3.85 TeraFLOPs.

4.4 Discussion

The GPGPU-accelerated implementation was applied to solve the 2D advection-reaction-diffusion equation for two general situations: (i) when the parameters of the model are constant and (ii) when the parameters of the model are time-varying. In the first case, it was possible to calculate the solution considering mesh grids containing up to 20 million grid points, as shown in Table 4.3. Considering an standard Landsat image spatial resolution of 30×30 meters, this mesh grid size would be roughly equivalent to an area of 18,000 km². Remember that the simulations were performed using an inexpensive office computer equipped with a NVIDIA GTX 1060 GPGPU card. This result illustrates that, besides improved performance, as shown in subsection 4.3.3, GPGPUs enable calculations to be performed with fine resolution over large areas. In the second case, considering the solution with time-varying parameters, the recalculation of the matrices causes extra computational burden, as shown in Fig. 4.13. In this case, the largest grid point for which the solution was delivered without a performance drop was about 4 million grid points, as shown in Table 4.4. In this case, considering an standard Landsat image spatial resolution of 30×30 meters, the mesh grid size is equivalent to an area of roughly 4,000 km². In both cases, the use of the GPGPU-accelerated implementation presented in this paper enabled the solution of environmental problems using a fine resolution over areas spanning several kilometers.

4.5 Final remarks

In this paper, we presented a GPGPU-accelerated implementation of the solution for the 2D advection-reaction-diffusion equation. The implementation was shown to be general and to allow a number of particular cases, such as the modelling of air pollution concentration, the regeneration of forest and the groundwater flow in unconfined aquifers.

Future research should be dedicated to the design and implementation of an user-friendly interface for the 2D GPGPU-accelerated implementation of the ARD equation. This will enable more widespread use of the implementation. The entire code for the implementation can be freely downloaded from `http://modelagemambientaluffs.blogspot.com.br/` [78].

Chapter 5

Final remarks

In this master thesis, I presented three studies based on computational models developed using parallel computing in GPGPU with CUDA technology. The first study (Chapter 2) showed that the application of GPGPU in the modeling of groundwater flow problems in heterogeneous and anisotropic unconfined aquifers allows to obtain superior performances to the models developed by approaches based on sequential computation. The results demonstrated that the parallel implementation significantly reduces the simulation time since it provided an acceleration of up to 56 times in the solution process of the groundwater flow equation.

The second study (Chapter 3) presented a methodology applicable as an auxiliary evaluation and design tool for forest regeneration and regeneration modeling projects. This study showed the effectiveness of the use of artificial neural network sets for the estimation of band widths of riparian protection in the provision of appropriate nutrient absorption conditions. This study also demonstrated the applicability of diffusive logistic growth models to simulate forest regeneration processes, using GPGPU in the simulation process, presenting results that confirm the validity of this methodology for environmental management and conservation decision-making.

The third study (Chapter 4) presented a GPGPU implementation for the solution of the 2D advection-reaction-diffusion equation that has been shown to be general enough to allow the modeling of a series of particular cases, such as atmospheric pollutant dispersion, forest growth and regeneration and groundwater modeling. Future studies in the topic shall cover the development of a computational tool that can provide a friendly graphical interface to end-users, in order to allow widespread use of the computational models implemented in this research. Further, we encourage further development of the models such that they contemplate new cases such as:

- the study of water injection and extraction in wells, for confined and unconfined aquifers;
- dispersion of pollutants in the soil;

- regeneration of forests considering interfering agents, by means of a decay term (simulating the degradation and regeneration process);
- dispersion of air pollutants using area and volumetric sources;
- pollution scattering studies with obstacles (buildings or containment barriers);

In addition to the development of new applications, it is recommended that in future work the analysis of the numerical stability of the models be developed together with convergence studies and new approaches to the evaluation of performance of GPGPU cards. Other aspects to be studied are: analysis of uncertainties in the parameters and analysis of uncertainties in the conceptual and numerical model together with test with different calibration methods.

Bibliography

[1] CARLOTTO, T.; SILVA, R. V. da; GRZYBOWSKI, J. M. A GPGPU-accelerated implementation of groundwater flow model in unconfined aquifers for heterogeneous and anisotropic media. *Environmental Modelling and Software*, Elsevier Ltd, v. 101, p. 64–72, 2018. ISSN 13648152. Disponível em: <https://doi.org/10.1016/j.envsoft.2017.12.004>.

[2] RICHIT, L. et al. Modelling forest regeneration for performance-oriented riparian buffer strips. *Ecological Engineering*, v. 106, n. Part A, p. 308 – 322, 2017. ISSN 0925-8574. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925857417303178>.

[3] SANTIN, F.; SILVA, R. da; GRZYBOWSKI, J. Artificial neural network ensembles and the design of performance-oriented riparian buffer strips for the filtering of nitrogen in agricultural catchments. *Ecological Engineering*, v. 94, p. 493 – 502, 2016. ISSN 0925-8574. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0925857416303330>.

[4] SURVEY, U. G. Landsat scene id: Lt52220792000038cub00. 2000–02–07.

[5] SURVEY, U. G. Landsat scene id: Lt52220792010049cub00. 2010–02–18.

[6] SURVEY, U. G. Landsat scene id: Lc82220792016018lgn00. 2016–01–18.

[7] SHANNON, K. et al. *firelab/windninja: 3.3.1*. 2017. Disponível em: <https://doi.org/10.5281/zenodo.848688>.

[8] WAINWRIGHT, J.; MULLIGAN, M. *Environmental Modelling Finding Simplicity in Complexity*. West Sussex: John Wiley & Sons, 2004. 408 p. ISBN 0471496170.

[9] ZOLNERKEVIC, I. *Desafios em águas profundas*. São Paulo, 2012. 170–173 p. Disponível em: <http://revistapesquisa.fapesp.br/wp-content/uploads/2012/08/170-173_aguas_profundas.pdf?cffa13>.

[10] USGS. *Aquifers and Groundwater, from USGS Water-Science School*. Disponível em: <http://water.usgs.gov/edu/earthgwaquifer.html>.

[11] SHARMA, Y.; KULKARNI, D. B. GPU Based Acceleration of WRF Model : A Review. *International Journal of Science and Research*, v. 4, n. 7, p. 2013–2016, 2015.

[12] NEWMAN, E. I. Water Movement Through Root Systems. *Philosophical Transactions of the Royal Society B: Biological Sciences*, v. 273, n. 927, p. 463–478, 1976. ISSN 0962-8436.

[13] FARBER, R. *CUDA Application Design and Development*. [S.l.: s.n.], 2011. ISBN 9780123884268.

[14] GALIANO, V. et al. GPU-based parallel algorithms for sparse nonlinear systems . *J. Parallel Distrib. Comput.*, Elsevier Inc., v. 72, n. 9, p. 1098–1105, 2012. ISSN 0743-7315. Disponível em: <http://dx.doi.org/10.1016/j.jpdc.2011.10.016>.

[15] ALFONSI, G. et al. GPGPU implementation of mixed spectral-finite difference computational code for the numerical integration of the three-dimensional time-dependent incompressible Navier-Stokes equations. *Computers and Fluids*, Elsevier Ltd, v. 102, p. 237–249, 2014. ISSN 00457930. Disponível em: <http://dx.doi.org/10.1016/j.compfluid.2014.07.005>.

[16] JI, X. et al. Parallelization of MODFLOW using a GPU library. *Groundwater*, v. 52, n. 4, p. 618–623, 2014. ISSN 17456584.

[17] LE, P. V. V. et al. Environmental Modelling & Software GPU-based high-performance computing for integrated surface e sub-surface fl ow modeling. *Environmental Modelling and Software*, Elsevier Ltd, v. 73, p. 1–13, 2015. ISSN 1364-8152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2015.07.015>.

[18] CRUZ, R. de la et al. Optimization of atmospheric transport models on HPC platforms. *Computers and Geosciences*, Elsevier, v. 97, p. 30–39, 2016. ISSN 00983004. Disponível em: <http://dx.doi.org/10.1016/j.cageo.2016.08.019>.

[19] MIELIKAINEN, J. et al. Compute unified device architecture (CUDA)-based parallelization of WRF Kessler cloud microphysics scheme. *Computers and Geosciences*, v. 52, p. 292–299, 2013. ISSN 00983004.

[20] ROUHOLAHNEJAD, E. et al. A parallelization framework for calibration of hydrological models. *Environmental Modelling and Software*, Elsevier Ltd, v. 31, p. 28–36, 2012. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2011.12.001>.

[21] JI, X.; CHENG, T.; WANG, Q. CUDA-based solver for large-scale groundwater flow simulation. *Engineering with Computers*, v. 28, n. 1, p. 13–19, 2012. ISSN 01770667.

[22] HWANG, H. T. et al. A parallel computational framework to solve flow and transport in integrated surface-subsurface hydrologic systems. *Environmental Modelling and Software*, Elsevier Ltd, v. 61, p. 39–58, 2014. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2014.06.024>.

[23] SU, D.; Ulrich Mayer, K.; MACQUARRIE, K. T. Parallelization of MIN3P-THCm: A high performance computational framework for subsurface flow and reactive transport simulation. *Environmental Modelling and Software*, Elsevier Ltd, v. 95, p. 271–289, 2017. ISSN 13648152. Disponível em: <http://dx.doi.org/10.1016/j.envsoft.2017.06.008>.

[24] ACEVEDO, M. A.; MARCANO, M.; FLETCHER, R. J. A diffusive logistic growth model to describe forest recovery. *Ecological Modelling*, Elsevier B.V., v. 244, p. 13–19, 2012. ISSN 03043800. Disponível em: <http://dx.doi.org/10.1016/j.ecolmodel.2012.07.012>.

[25] LEELÕSSY, Á. et al. Dispersion modeling of air pollutants in the atmosphere : a review. *Central European Journal of Geosciences Dispersion*, v. 6, n. 3, 2014.

[26] PINHEIRO, A. et al. GPU-based implementation of a diagnostic wind field model used in real-time prediction of atmospheric dispersion of radionuclides. *Progress in Nuclear Energy*, Elsevier Ltd, v. 100, p. 146–163, 2017. ISSN 01491970. Disponível em: <http://dx.doi.org/10.1016/j.pnucene.2017.05.027>.

[27] FREEZE, A. R. Three-Dimensional, Transient, Saturated-Unsaturated Flow in a Ground-Water Basin. *Water Resources Research*, n. 2, p. 347–366, 1971.

[28] LANGE, W. J. et al. An operational, multi-scale, multi-model system for consensus-based, integrated water management and policy analysis: The Netherlands Hydrological Instrument. *Environmental Modelling & Software*, Elsevier Ltd, v. 59, p. 98–108, set. 2014. ISSN 13648152. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1364815214001406>.

[29] VOECKLER, H. M.; ALLEN, D. M.; ALILA, Y. Modeling coupled surface water Groundwater processes in a small mountainous headwater catchment. *Journal of Hydrology*, Elsevier B.V., v. 517, p. 1089–1106, set. 2014. ISSN 00221694. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0022169414004752>.

[30] PARLANGE, M. B.; ALBERTSON, J. D. Recession flow analysis for aquifer parameter determination. *Water Resources Research*, v. 34, n. 7, p. 1851–1857, 1998.

[31] CZARNECKI, J. B. et al. *Groundwater-Flow Model of the Ozark Plateaus Aquifer System , Northwestern Arkansas , Southeastern Kansas , Southwestern Missouri , and Northeastern Oklahoma*. Reston,Virginia, 2010.

[32] GRAAF, I. et al. Dynamic attribution of global water demand to surface water and groundwater resources: Effects of abstractions and return flows on river discharges. *Advances in Water Resources*, Elsevier Ltd, v. 64, p. 21–33, fev. 2014. ISSN 03091708. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0309170813002467>.

[33] HARBAUGH, A. W. *MODFLOW-2005 , The U . S . Geological Survey Modular Ground-Water Model-the Ground-Water Flow Process*. [S.l.], 2005.

[34] MARKSTROM, S. et al. GSFLOW: Coupled Ground-Water and Surface Water Flow Model Based on the Integration of the Precipitation-Runoff Modeling System (PRMS) and the Modular Ground Water Flow Model (MODFLOW-2005). *US Geological Survey Techniques and Methods*, 2008. Disponível em: <http://pubs.usgs.gov/tm/tm6d1/>.

[35] HEATH, R. C. *Basic Ground-Water Hydrology*. Reston,Virginia: U.S.Geological Survey, 1983. 86 p. ISSN <null>. ISBN 0607689730.

[36] PHILLIPS, J. Nonpoint source pollution control effectiveness of riparian forests along a coastal plain river. *Journal of Hydrology*, v. 110, p. 221–237, 1989.

[37] MANDER, U. et al. Efficiency and dimensioning of riparian buffer zones in agricultural catchments. *Ecological Engineering*, v. 8, p. 299–324, 1997.

[38] SYVERSEN, N. Effect of buffer zones in the Nordic climate: the influence of width, amount of surface runoff, seasonal variation and vegetation type on retention efficiency for nutrient and particle runoff. *Ecological Engineering*, v. 24, p. 483–490, 2005.

[39] MAYER, P. M.; REYNOLDS, S. K.; CANFIELD, T. J. *Riparian buffer width, vegetative cover, and nitrogen removal effectiveness: a review of current science and regulations*. [S.l.], 2005. 1–40 p. Disponível em: <http://nepis.epa.gov/Exe/ZyPDF.cgi/2000O182.PDF?Dockey=2000O182.PDF>.

[40] CORRELL, D. Principles of planning and establishment of buffer zones. *Ecological Engineering*, v. 24, p. 433–439, 2005.

[41] ZAHAWI, R. A. et al. Using lightweight unmanned aerial vehicles to monitor tropical forest recovery. *Biological Conservation*, v. 186, p. 287 – 295, 2015. ISSN 0006-3207. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0006320715001421>.

[42] SAHU, M.; GU, R. Modeling the effects of riparian buffer zone and contour strips on stream water quality. *Ecological Engineering*, v. 35, p. 1167–1177, 2009.

[43] MANDER, U.; HAYAKAWA, Y.; KUUSEMETS, V. Purification processes, ecological functions, planning and design of riparian buffer zones in agriculturas watersheds. *Ecological Engineering*, v. 24, p. 421–432, 2005.

[44] Le Phong, V. V. et al. GPU-based high-performance computing for integrated surface sub-surface flow modeling. *Environmental Modelling & Software*, Elsevier Ltd, v. 73, p. 1–13, 2015. ISSN 13648152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815215300207>.

[45] MOLNÁR, F. et al. Simulation of reaction-diffusion processes in three dimensions using CUDA. *Chemometrics and Intelligent Laboratory Systems*, Elsevier B.V., v. 108, n. 1, p. 76–85, 2011. ISSN 01697439. Disponível em: <http://dx.doi.org/10.1016/j.chemolab.2011.03.009>.

[46] KAJISHIMA, T.; TAIRA, K. *Computational Fluid Dynamics: Incompressible Turbulent Flows*. Springer International Publishing, 2016. ISBN 9783319453040. Disponível em: <https://books.google.com.br/books?id=3QUtDQAAQBAJ>.

[47] ROUHOLAHNEJAD, E. et al. A parallelization framework for calibration of hydrological models. *Environmental Modelling & Software*, v. 31, p. 28 – 36, 2012. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815211002829>.

[48] ZHANG, X. et al. Efficient multi-objective calibration of a computationally intensive hydrologic model with parallel computing software in Python. *Environmental Modelling & Software*, v. 46, p. 208 – 218, 2013. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815213000765>.

[49] HWANG, H.-T. et al. A parallel computational framework to solve flow and transport in integrated surface-subsurface hydrologic systems. *Environmental Modelling & Software*, v. 61, p. 39 – 58, 2014. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815214001972>.

[50] BOTROS, F. E. et al. Richards Equation-Based Modeling to Estimate Flow and Nitrate Transport in a Deep Alluvial Vadose Zone. *Vadose Zone Journal*, v. 11, 2012. ISSN 1539-1663.

[51] FREEZE, R. A.; WITHERSPOON, P. A. Theoretical analysis of regional groundwater flow: 1. analytical and numerical solutions to the mathematical model. *Water*

Resources Research, v. 2, n. 4, p. 641–656, 1966. ISSN 1944-7973. Disponível em: <http://dx.doi.org/10.1029/WR002i004p00641>.

[52] HE, B. et al. Estimating monthly total nitrogen concentration in streams by using artificial neural network. *Journal of Environmental Management*, v. 92, n. 1, p. 172 – 177, 2011. ISSN 0301-4797. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0301479710003002>.

[53] LAMB, R.; BEVEN, K. J. Using interactive recession curve analysis to specify a general catchment storage model. *Hydrology and Earth System Sciences*, v. 1, p. 101–113, 1997. ISSN 1607-7938. Disponível em: <http://www.hydrol-earth-syst-sci.net/1/101/1997/hess-1-101-1997.html>.

[54] LYNE, V.; HOLLICK, M. Stochastic time-variable rainfall-runoff modelling. In: *Institute of Engineers Australia National Conference*. [S.l.: s.n.], 1979. v. 79, n. 10, p. 89–93.

[55] MILLER, C. T. et al. Numerical simulation of water resources problems: Models, methods, and trends. *Advances in Water Resources*, v. 51, p. 405 – 437, 2013. ISSN 0309-1708. 35th Year Anniversary Issue. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0309170812001431>.

[56] MENDOZA, G.; MARTINS, H. Multi-criteria decision analysis in natural resource management: A critical review of methods and new modelling paradigms. *Forest Ecology and Management*, v. 230, n. 1-3, p. 1 – 22, 2006. ISSN 0378-1127. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378112706002258>.

[57] NICKOLLS, J. et al. Scalable parallel programming with CUDA. *Queue*, ACM, New York, NY, USA, v. 6, n. 2, p. 40–53, mar. 2008. ISSN 1542-7730. Disponível em: <http://doi.acm.org/10.1145/1365490.1365500>.

[58] SINGH, B. et al. Accelerating urban fast response lagrangian dispersion simulations using inexpensive graphics processor parallelism. *Environmental Modelling & Software*, v. 26, n. 6, p. 739 – 750, 2011. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815210003452>.

[59] ZHOU, J. et al. Multi-gpu implementation of a 3d finite difference time domain earthquake code on heterogeneous supercomputers. *Procedia Computer Science*, v. 18, p. 1255 – 1264, 2013. ISSN 1877-0509. 2013 International Conference on Computational Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050913004353>.

[60] JI, X. et al. Parallelization of MODFLOW using a GPU library. *Groundwater*, v. 52, n. 4, p. 618–623, 2014.

[61] BARRETT, R. et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. Philadelphia, PA: SIAM, 1994.

[62] FORMAGGIA, L.; SALA, M.; SALERI, F. Domain decomposition techniques. In: ____. *Numerical Solution of Partial Differential Equations on Parallel Computers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 135–163. ISBN 978-3-540-31619-0.

[63] NAKAJIMA, K. Large-scale simulations of 3D groundwater flow using parallel geometric multigrid method. *Procedia Computer Science*, v. 18, p. 1265 – 1274, 2013. ISSN 1877-0509. 2013 International Conference on Computational Science. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050913004365>.

[64] WU, Y. et al. Parallelization of a hydrological model using the message passing interface. *Environmental Modelling & Software*, v. 43, p. 124 – 132, 2013. ISSN 1364-8152. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1364815213000327>.

[65] MCDONALD, M. G.; HARBAUGH, A. W. The history of MODFLOW. *Ground water*, v. 41, n. 2, p. 280–283, 2003. ISSN 0017-467X.

[66] CRANK, J.; NICOLSON, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, v. 43, n. 01, p. 50–67, 1947. ISSN 1019-7168. Disponível em: <http://www.journals.cambridge.org/abstract_S0305004100023197>.

[67] CRANK, J.; NICOLSON, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, v. 6, n. 1, p. 1–7, 1996. ISSN 1572-9044. Disponível em: <http://dx.doi.org/10.1007/BF02127704>.

[68] BELL, N.; GARLAND, M. CUSP: generic parallel algorithms for sparse matrix and graph computations, version 0.3.0. *Proc. of the 1st Int. Conference on Emerging Trends in Mechanical Engineering - ICETME 2013*, 2013.

[69] BOUSSINESQ, J. Recherches théoriques sur l'écoulement des nappes d'eau infiltrées dans le sol et sur le débit des sources. *Journal de Mathématiques Pures et Appliquées*, v. 10, p. 5–78, 1904. Disponível em: <http://eudml.org/doc/235283>.

[70] DELLEUR, J. *The Handbook of Groundwater Engineering, Second Edition*. CRC Press, 2006. ISBN 9781420006001. Disponível em: <https://books.google.com.br/books?id=4WDLBQAAQBAJ>.

[71] FETTER, C. *Applied Hydrogeology*. Prentice Hall, 2001. ISBN 9780130882394. Disponível em: <https://books.google.ca/books?id=1kS9QgAACAAJ>.

[72] CLEARY, R. W. Águas Subterrâneas. p. 117, 1989. Disponível em: <http://www.clean.com.br/cleary.pdf>.

[73] CHAPMAN, T. G. Modeling Groundwater Flow Over Sloping Beds. *Water Resources Research*, v. 16, n. 6, p. 1114–1118, 1980. Disponível em: <http://onlinelibrary.wiley.com/doi/10.1029/WR016i006p01114/abstract>.

[74] GUO, W. Transient groundwater flow between reservoirs and water table aquifers. *Journal of Hydrology*, v. 195, p. 370–384, 1997.

[75] ESSINK, G. O. *Groundwater Modelling*. Utrecht: Utrecht University, 2000.

[76] TEAM, M. V. S. *Visual Studio Community 2013*. 2017. Disponível em: <https://www.visualstudio.com/en-us/news/releasenotes/vs2013-community-vs>.

[77] ZONE, N. D. *CUDA Toolkit Documentation 7.5*. 2017. Disponível em: <http://docs.nvidia.com/cuda/>.

[78] UFFS, E. M. at. *CUDA-accelerated solution of the groundwater flow model in unconfined aquifers for heterogeneous and anisotropic media. Available at http://modelagemambientaluffs.blogspot.com.br.* 2017. Disponível em: <http://modelagemambientaluffs.blogspot.com.br/>.

[79] VORST, H. V. D.; DEKKER, K. Conjugate gradient type methods and preconditioning. *Journal of Computational and Applied Mathematics*, v. 24, p. 73–87, 1988.

[80] DECIAN, V. Análise e zoneamento da área de proteção ambiental dos rios Ligeirinho e Leãozinho (Erechim - RS). *PhD Thesis*, PPGERN, p. Federal University of São Carlos – Sao Paulo – SP, 2012.

[81] CPRM - Serviço Geológico do Brasil. *SIAGAS*. 2017. Disponível em: <http://siagasweb.cprm.gov.br/layout/index.php>.

[82] SAULNIER, G. M.; BEVEN, K.; OBLED, C. Including spatially variable effective soil depths in TOPMODEL. *Journal of Hydrology*, v. 202, n. 1-4, p. 158–172, 1997. ISSN 00221694.

[83] KER, J. C. Latossolos do Brasil Uma revisão. *Geonomos*, v. 5, n. 1, p. 17–40, 1998. Disponível em: <http://www.igc.ufmg.br/geonomos/PDFs/5_1_17_40_Ker.pdf>.

[84] MUNIZ, M. et al. Updated Brazilians Georeferenced Soil Database - An Improvement for International Scientific Information Exchanging. In: *Principles, Application and Assessment in Soil Science*. InTech, 2011. cap. 16, p. 309–332. ISBN 978-953-307-740-6. Disponível

em: <http://www.intechopen.com/books/principles-application-and-assessment-in-soil-science/updated-brazilian-s-georeferenced-soil-database-an-improvement-for-international-scientific-informat>.

[85] MAITRE, D. C. L.; SCOTT, D. F.; COLVIN, C. A review information on interactions between vegetation and groundwater. *Water SA*, v. 25, n. 2, p. 959–962, 2000. ISSN 03784738. Disponível em: <http://www.wrc.org.za/Knowledge Hub Documents/Water SA Journals/Manuscripts/1999/02/WaterSA_1999_02_apr99_p137.pdf>.

[86] RIDOLFI, L.; D'ODORICO, P.; LAIO, F. Effect of vegetation-water table feedbacks on the stability and resilience of plant ecosystems. *Water Resources Research*, v. 42, n. 1, p. 1–5, 2006. ISSN 00431397.

[87] SUJONO, J.; SHIKASHO, S.; HIRAMATSU, K. A comparison of techniques for hydrograph recession analysis. *Hydrological Processes*, v. 18, n. October 2002, p. 403–413, 2004. ISSN 08856087.

[88] EMBRAPA; IBGE. *Tipos de solos do Brasil*. 2011. Disponível em: <http://www.dpi.inpe.br/Ambdata/mapa_solos.php>.

[89] GONÇALVES, A.; LIBARDI, P. Análise da determinação da condutividade hidraulica do solo pelo Método do Perfil Instantâneo. *Revista Brasileira de Ciencias do Solo*, v. 37, p. 1174–1184, 2013.

[90] AGENCY, E. E. P. *Riparian buffer width, vegetative cover and nitrogen removal effectiveness: a review of current science and regulations*. [S.l.]: National Risk Management Research Laboratory, 2005.

[91] BOZ, B.; GUMIERO, B. Nitrogen removal in an afforested riparian zone: the contribution of denitrification processes. *Hydrobiologia*, Springer, v. 774, n. 1, p. 167–182, 2016.

[92] PINAY, G.; ROQUES, L.; FABRES, A. Spatial and temporal patterns of denitrification in a riparian forest. *Journal of Applied Ecology*, v. 30, p. 581–591, 1993.

[93] GUNDERSEN, P. Nitrogen deposition and the forest nitrogen cycle: role of denitrification. *Forest Ecology and Management*, Elsevier, v. 44, n. 1, p. 15–28, 1991.

[94] BOTHE, H.; FERGUSON, S.; NEWTON, W. E. *Biology of the nitrogen cycle*. [S.l.]: Elsevier, 2006.

[95] DENDOOVEN, L.; ANDERSON, J. Dynamics of reduction enzymes involved in the denitrification process in pasture. *Soil Biology and Biochemistry*, v. 26, p. 1501–1506, 1994.

[96] PINAY, G. et al. Geomorphic control of denitrification in large river floodplain soils. *Biogeochemistry*, v. 30, p. 9–29, 2000.

[97] GROFFMAN, P. M. et al. Denitrification in grass and forest vegetated filter strips. *Journal of environmental Quality*, American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America, v. 20, n. 3, p. 671–674, 1991.

[98] ROBERTSON, G. P.; TIEDJE, J. M. Deforestation alters denitrification in a lowland tropical rain forest. *Nature*, Nature Publishing Group, v. 336, n. 6201, p. 756–759, 1988.

[99] HILL, A. R. et al. Subsurface denitrification in a forest riparianzone: interactions between hydrology and supplies ofnitrate and organic carbon. *Biogeochemistry*, Springer, v. 51, n. 2, p. 193–223, 2000.

[100] LOWRANCE, R. Groundwater nitrate and denitrification in a coastal plain riparian forest. *Journal of environmental Quality*, American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America, v. 21, n. 3, p. 401–405, 1992.

[101] NEWCOMER, T. A. et al. Influence of natural and novel organic carbon sources on denitrification in forest, degraded urban, and restored streams. *Ecological Monographs*, Wiley Online Library, v. 82, n. 4, p. 449–466, 2012.

[102] LOWRANCE, R.; VELLIDIS, G.; HUBBARD, R. K. Denitrification in a restored riparian forest wetland. *Journal of Environmental Quality*, American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America, v. 24, n. 5, p. 808–815, 1995.

[103] PARKYN, S. Review of riparian buffer zone effectiveness. *US Ministry of Agriculture and Forestry*, v. 05, 2004.

[104] KUGLEROVA, L. et al. Towards optimizing riparian buffer zones: ecological and biogeochemical implications for forest management. *Forest Ecology and Management*, v. 334, p. 74–84, 2014.

[105] COLE, L.; BHAGWAT, S.; WILLIS, K. Recovery and resilience of tropical forests after disturbance. *Nature Communications*, v. 5, n. 3906, p. 207–226, 2014.

[106] CROOKSTON, N. L.; DIXON, G. E. The forest vegetation simulator: A review of its structure, content, and applications. *Computers and Electronics in Agriculture*, v. 49, n. 1, p. 60 – 80, 2005. ISSN 0168-1699. Decision Support Systems for Forest ManagementDecision Support in Multiple purpose Forestry. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169905000347>.

[107] CROOKSTON, N. et al. Addressing climate change in the Forest Vegetation Simulator to asses impacts on landscape forest dynamics. *USDA Forest Service Proceedings RMRS-P*, v. 61, 2010.

[108] USDA; CENTER, U. F. S. C. C. R. *Forest Vegetation Simulator (FVS)*. 2010. Disponível em: <http://http://www.fs.usda.gov/ccrc/tools/fvs>.

[109] ACEVEDO, M. A.; MARCANO, M.; JR., R. J. F. A diffusive logistic growth model to describe forest recovery. *Ecological Modelling*, v. 244, p. 13 – 19, 2012. ISSN 0304-3800. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0304380012003390>.

[110] TWERY, M. J. et al. Ned-1: integrated analyses for forest stewardship decisions. *Computers and Electronics in Agriculture*, v. 27, n. 1-3, p. 167 – 193, 2000. ISSN 0168-1699. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169900001071>.

[111] TWERY, M. J. et al. Ned-2: A decision support system for integrated forest ecosystem management. *Computers and Electronics in Agriculture*, v. 49, n. 1, p. 24 – 43, 2005. ISSN 0168-1699. Decision Support Systems for Forest ManagementDecision Support in Multiple purpose Forestry. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169905000487>.

[112] BARCZI, J.-F. et al. Amapsim: a structural whole-plant simulator based on botanical knowledge and designed to host external functional models. *Annals of Botany*, Annals Botany Co, v. 101, n. 8, p. 1125–1138, 2008.

[113] TWILLEY, R. R. et al. Adapting an ecological mangrove model to simulate trajectories in restoration ecology. *Marine Pollution Bulletin*, Elsevier, v. 37, n. 8, p. 404–419, 1999.

[114] COLIGNY, F. D. et al. Capsis: computer-aided projection for strategies in silviculture: advantages of a shared forest-modelling platform. In: *International Workshop of IUFRO working party*. [S.l.: s.n.], 2003. v. 4, p. 2–5.

[115] COLIGNY, F. de. Efficient building of forestry modelling software with the capsis methodology. In: IEEE. *Plant Growth Modeling and Applications, 2006. PMA'06. Second International Symposium on*. [S.l.], 2006. p. 216–222.

[116] GOURLET-FLEURY, S. et al. Modelling forest dynamics for practical management purposes. *Bois et forêts des tropiques*, n. 280, p. 41–52, 2004.

[117] GOURLET-FLEURY, S. Individual-based spatially explicit modelling of forest stands in french guiana. European Union, 1999.

[118] GOURLET-FLEURY, S. et al. Using models to predict recovery and assess tree species vulnerability in logged tropical forests: a case study from french guiana. *Forest ecology and management*, Elsevier, v. 209, n. 1, p. 69–85, 2005.

[119] PRETZSCH, H.; BIBER, P.; ĎURSKỲ, J. The single tree-based stand simulator silva: construction, application and evaluation. *Forest ecology and management*, Elsevier, v. 162, n. 1, p. 3–21, 2002.

[120] PORTÉ, A.; BARTELINK, H. Modelling mixed forest growth: a review of models for forest management. *Ecological modelling*, Elsevier, v. 150, n. 1, p. 141–188, 2002.

[121] DIAZ-BALTEIRO, L.; ROMERO, C. Forest management optimisation models when carbon captured is considered: a goal programming approach. *Forest Ecology and Management*, v. 174, n. 1-3, p. 447 – 457, 2003. ISSN 0378-1127. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378112702000750>.

[122] BAGDON, B. A.; HUANG, C.-H.; DEWHURST, S. Managing for ecosystem services in northern arizona ponderosa pine forests using a novel simulation-to-optimization methodology. *Ecological Modelling*, v. 324, p. 11 – 27, 2016. ISSN 0304-3800. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0304380015005803>.

[123] GARDNER, M.; DORLING, S. Artificial neural networks (the multilayer perceptron) - a review of applications in the atmospheric sciences. *Atmospheric Environment*, v. 12, n. 14/15.

[124] MAIER, H.; DANDY, G. Neural network based modeling of environmental variables: a systematic approach. *Mathematical and Computer Modelling*, v. 33, p. 669–682, 2001.

[125] DALIAKOPOULOS, I. N.; COULIBALY, P.; TSANIS, I. K. Groundwater level forecasting using artificial neural networks. *Journal of Hydrology*, v. 309, n. 1-4, p. 229–240, 2005. ISSN 0022-1694. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022169404005840>.

[126] LALLAHEM, S. et al. On the use of neural networks to evaluate groundwater levels in fractured media. *Journal of Hydrology*, v. 307, n. 1-4, p. 92 – 111, 2005. ISSN 0022-1694. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022169404004925>.

[127] GARCIA, L. A.; SHIGIDI, A. Using neural networks for parameter estimation in ground water. *Journal of Hydrology*, v. 318, n. 1-4, p. 215 – 231, 2006. ISSN 0022-1694. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022169405003057>.

[128] SAHOO, G. B. et al. Application of artificial neural networks to assess pesticide contamination in shallow groundwater. *Science of The Total Environment*, v. 367, n. 1, p. 234 – 251, 2006. ISSN 0048-9697. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0048969705008910>.

[129] TOSH, C.; RUXTON, G. The need for stochastic replication of ecological neural networks. *Phylosophical Transactions of the Royal Society B*, v. 362, p. 455–460, 2007.

[130] YOON, H. et al. A comparative study of artificial neural networks and support vector machines for predicting groundwater levels in a coastal aquifer. *Journal of Hydrology*, v. 396, n. 1-2, p. 128 – 138, 2011. ISSN 0022-1694. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022169410006761>.

[131] HOLZBECHER, E. *Environmental Modeliing using MATLAB(C)*. [S.l.]: Springer - Verlage Berlin Heidelberg, 2007.

[132] SKELLAM, J. G. Random dispersal in theoretical populations. *Biometrika*, JSTOR, v. 38, n. 1/2, p. 196–218, 1951.

[133] EDELSTEIN-KESHET, L. *Mathematical models in biology*. [S.l.]: Siam, 1988. v. 46.

[134] OKUBO, A.; LEVIN, S. A. *Diffusion and ecological problems: modern perspectives*. [S.l.]: Springer Science & Business Media, 2013. v. 14.

[135] VERHULST, P. Recherches mathématiques sur la loi d'accroissement de la population. *Nouv. Mém. de l'Académie Royale des Sci. et Belles-Lettres de Bruxelles*, v. 18, p. 1–41, 1845.

[136] VERHULST, P. Deuxiéme memoire sur la loi d'accroissement de la population. *Mém. de l'Académie Royale des Sci. et Belles-Lettres de Bruxelles*, v. 20, p. 1–32, 1847.

[137] MELICA, V.; INVERNIZZI, S.; CARISTI, G. Logistic density-dependent growth of an aurelia aurita polyps population. *Ecological Modelling*, v. 291, p. 1 – 5, 2014. ISSN 0304-3800. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0304380014003366>.

[138] NASELL, I. Extinction and quasi-stationarity in the verhulst logistic model. *Journal of Theoretical Biology*, v. 211, n. 1, p. 11 – 27, 2001. ISSN 0022-5193. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022519301923288>.

[139] PELEG, M.; CORRADINI, M. G.; NORMAND, M. D. The logistic (verhulst) model for sigmoid microbial growth curves revisited. *Food Research Inter-*

national, v. 40, n. 7, p. 808 – 818, 2007. ISSN 0963-9969. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0963996907000269>.

[140] ROSEN, G. Characterizing conditions for generalized verhulst logistic growth of a biological population. *Bulletin of Mathematical Biology*, v. 46, n. 5, p. 963 – 965, 1984. ISSN 0092-8240. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0092824084800142>.

[141] HOLMES, E. E. et al. Partial differential equations in ecology: spatial interactions and population dynamics. *Ecology*, Wiley Online Library, v. 75, n. 1, p. 17–29, 1994.

[142] NVIDIA. *CUDA Technology*. 2007. Disponível em: <http://www.nvidia.com/CUDA>.

[143] GRASS, D. T. *Geographic Resources Analysis Support System (GRASS)*. GNU General Public License, 2014. Disponível em: <http://grass.osgeo.org>.

[144] SOLANO, R. et al. *MODIS Vegetation Index User's Guide*. [S.l.]: Vegetation Index and Phenology Lab - University of Arizona, 2010.

[145] VALERIANO, M. M. *Modelos digitais de elevação de microbacias elaborados com krigagem*. São Jose dos Campos, SP, 2002. 54 p.

[146] VALERIANO, M. M. *Modelo digital de elevação com dados SRTM disponíveis para a América do Sul*. São Jose dos Campos, SP, 2004. 72 p.

[147] MONTGOMERY, D. R.; DIETRICH, W. E. Source Areas, drainage density, channel initiation. *Water Resources Research*, v. 25, n. 8, p. 1907–1918, 1989.

[148] MONTGOMERY, D. R.; DIETRICH, W. E. Channel initiation and the problem of landscape scale. *Science*, v. 255, p. 826–830, 1992. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/17756428>.

[149] SURVEY, U. G. Landsat scene id: Lc82220792014348lgn00. 2014–12–14.

[150] HARMEL, D. et al. Compilation of measured nutrient data for agricultural land uses in the united states. *Journal of the American Water Resources Association*, October, p. 1163–1178, 2006.

[151] ANA. *Hidroweb - Sistema de informações hidrológicas*. 2015. Disponível em: <http://www.snirh.gov.br/hidroweb/>.

[152] NETCH-NIELSEN, R. Kolmogorov's mapping neural network existence theorem. *First IEEE International Joint Conference on Neural Networks*, p. 11–14, 1987.

[153] HAYKIN, S. *Neural networks - a comprehensive foundation - 2nd edition*. [S.l.]: Prentice-Hall, 1999.

[154] CHELLAPILA, K.; FOGEL, D. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE. Transactions on Neural Networks*, v. 10, n. 6, 1999.

[155] DIETZEL, M. et al. Artificial neural networks for differential diagnosis of breast lesions in mr-mammography: A systematic approach addressing the influence of network architecture on diagnostic performance using a large clinical database. *European Journal of Radiology*, v. 81, n. 7, p. 1508 – 1513, 2012. ISSN 0720-048X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0720048X11002944>.

[156] GOPINATH, P.; REDDY, N. Toward intelligent web monitoring: performance of committee neural network vs. single neural network. In: . [S.l.: s.n.], 2000.

[157] LISBOA, P. J.; TAKTAK, A. F. The use of artificial neural networks in decision support in cancer: A systematic review. *Neural Networks*, v. 19, n. 4, p. 408 – 415, 2006. ISSN 0893-6080. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0893608005002844>.

[158] TARBOTON, D. G. Fractals, scaling and nonlinear variability in hydrology fractal river networks, horton's laws and tokunaga cyclicity. *Journal of Hydrology*, v. 187, n. 1, p. 105 – 117, 1996. ISSN 0022-1694. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022169496030892>.

[159] CAMPOS, D.; FORT, J.; MÉNDEZ, V. Transport on fractal river networks: Application to migration fronts. *Theoretical Population Biology*, v. 69, n. 1, p. 88 – 93, 2006. ISSN 0040-5809. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0040580905001036>.

[160] RAMBO, B. A fisionomia do Rio Grande do Sul, 2a. *Porto Alegre: Ed. Livraria Selbach*, 1956.

[161] SOUZA, A. F. Ecological interpretation of multiple population size structures in trees: the case of araucaria angustifolia in South America. *Austral Ecology*, Wiley Online Library, v. 32, n. 5, p. 524–533, 2007.

[162] GEIST, H.; LAMBIN, E. Proximate causes and underlying driving forces of tropical deforestation tropical forests are disappearing as the result of many pressures, both local and regional, acting in various combinations in different geographical locations. *BioScience*, Oxford University Press, v. 52, n. 2, p. 143–150, 2002.

[163] HUECK, K. Distribuição e habitat natural do Pinheiro do Paraná (Araucaria angustifolia). *Boletim da Faculdade de Filosofia, Ciências e Letras, Universidade de São Paulo. Botânica*, JSTOR, n. 10, p. 3–24, 1953.

[164] KLEIN, R. Árvores nativas da floresta subtropical do Alto Uruguai. *Sellowia*, 1972.

[165] BITTENCOURT, J. et al. Conservation, management and sustainable use of Araucaria angustifolia genetic resources in Brazil. *Org.). Challenges in managing forest genetic resource for livelihoods: examples from Argentina and Brazil. Roma: IPGRI*, p. 133–48, 2004.

[166] BITTENCOURT, J.; SEBBEN, A. Patterns of pollen and seed dispersal in a small, fragmented population of the wind-pollinated tree Araucaria angustifolia in Southern Brazil. *Heredity*, Nature Publishing Group, v. 99, n. 6, p. 580–591, 2007.

[167] STEFENON, V.; GAILING, O.; FINKELDY, R. Genetic structure of Araucaria angustifolia (araucariaceae) populations in Brazil: implications for the in situ conservation of genetic resources. *Plant Biology*, Georg Thieme Verlag Stuttgart KG· New York, v. 9, n. 04, p. 516–525, 2007.

[168] BACKES, P.; IRGANG, B. Árvores do sul: guia de identificação & interesse ecologico as principais espécies nativas sul-brasileiras. *Rio de Janeiro: Instituto Souza Cruz [2002] 326p.-col. illus.. Por Icones. Geog*, v. 4, 2002.

[169] BIRGAND, F. et al. Nitrogen removal in streams of agricultural catchments - a literature review. *Critical Reviews in Environmental Science and Technology*, Taylor & Francis, v. 37, n. 5, p. 381–487, 2007.

[170] BRASIL, C. Resolução. 357, de 17 de março de 2005. *Conselho Nacional do Meio Ambiente-CONAMA*, v. 357, 2005.

[171] PETERJOHN, W. T.; CORRELL, D. L. Nutrient dynamics in an agricultural watershed: observations on the role of a riparian forest. *Ecology*, Wiley Online Library, v. 65, n. 5, p. 1466–1475, 1984.

[172] SCHWER, C.; CLAUSEN, J. Vegetative filter treatment of dairy milkhouse wastewater. *Journal of Environmental Quality*, American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America, v. 18, n. 4, p. 446–451, 1989.

[173] MANDER, Ü. The renovation effect of polluted surface flow in vegetated buffer strips. *Acta Commentationes Universitatis Tartuensis*, v. 701, p. 77–89, 1985.

[174] MITSCH, W. J.; DORAGE, C. L.; WIEMHOFF, J. R. Ecosystem dynamics and a phosphorus budget of an alluvial cypress swamp in southern illinois. *Ecology*, Wiley Online Library, v. 60, n. 6, p. 1116–1124, 1979.

[175] KNAUER, N.; MANDER, Ü. Untersuchungen über die filterwirkung verschiedener saumbiotope an gewässern in schleswig-holstein. 1. mitteilung: Filterung von stickstoff und phosphor. *Zeitschrift für Kulturtechnik und Landentwicklung, H*, v. 30, p. 365–376, 1989.

[176] JACOBS, T.; GILLIAM, J. Riparian losses of nitrate from agricultural drainage waters. *Journal of environmental quality*, American Society of Agronomy, Crop Science Society of America, and Soil Science Society of America, v. 14, n. 4, p. 472–478, 1985.

[177] TAGUE, C.; BAND, L. Evaluating explicit and implicit routing for watershed hydro-ecological models of forest hydrology at the small catchment scale. *Hydrological Processes*, Wiley Online Library, v. 15, n. 8, p. 1415–1439, 2001.

[178] FOLEY, J. A. et al. An integrated biosphere model of land surface processes, terrestrial carbon balance, and vegetation dynamics. *Global Biogeochemical Cycles*, Wiley Online Library, v. 10, n. 4, p. 603–628, 1996.

[179] TIKTAK, A.; GRINSVEN, H. van. Review of sixteen forest-soil-atmosphere models. *Ecological modelling*, Elsevier, v. 83, n. 1-2, p. 35–53, 1995.

[180] GERTEN, D. et al. Terrestrial vegetation and water balance - hydrological evaluation of a dynamic global vegetation model. *Journal of Hydrology*, Elsevier, v. 286, n. 1, p. 249–270, 2004.

[181] MÓRICZ, N. et al. Water balance study of a groundwater-dependent oak forest. *Acta Silv. Ling. Hung*, v. 6, p. 49–66, 2010.

[182] SMERDON, B. D.; REDDING, T.; BECKERS, J. An overview of the effects of forest management on groundwater hydrology. *Journal of Ecosystems and Management*, v. 10, n. 1, 2009.

[183] PECK, A.; WILLIAMSON, D. Effects of forest clearing on groundwater. *Journal of Hydrology*, Elsevier, v. 94, n. 1, p. 47–65, 1987.

[184] SIVAPALAN, M.; RUPRECHT, J. K.; VINEY, N. R. Water and salt balance modelling to predict the effects of land-use changes in forested catchments. 1. small catchment water balance model. *Hydrological Processes*, Wiley Online Library, v. 10, n. 3, p. 393–411, 1996.

[185] JANUSZEWSKI, M.; KOSTUR, M. Accelerating numerical solution of stochastic differential equations with CUDA. *Computer Physics Communications*, Elsevier B.V., v. 181, n. 1, p. 183–188, 2010. ISSN 00104655. Disponível em: <http://dx.doi.org/10.1016/j.cpc.2009.09.009>.

[186] NVIDIA. *NVIDIA Next Generation CUDA Compute Architecture Fermi*. [S.l.], 2009. v. 1.1, 1–22 p.

[187] NVIDIA. *cuSPARSE*. 2015. Disponível em: <http://docs.nvidia.com/cuda/cusparse/>.

[188] PICKERING, B. P. et al. Directive-based GPU programming for computational fluid dynamics. *Computers and Fluids*, Elsevier Ltd, v. 114, p. 242–253, 2015. ISSN 00457930. Disponível em: <http://dx.doi.org/10.1016/j.compfluid.2015.03.008>.

[189] NVIDIA. *cuBLAS*. 2015. Disponível em: <http://docs.nvidia.com/cuda/cublas/index.html>.

[190] BARNEY, B. *Introduction to Parallel Computing*. 2016. Disponível em: <https://computing.llnl.gov/tutorials/parallel_comp/>.

[191] WALSH, S. D. et al. Accelerating geoscience and engineering system simulations on graphics hardware. *Computers and Geosciences*, Elsevier, v. 35, n. 12, p. 2353–2364, 2009. ISSN 00983004. Disponível em: <http://dx.doi.org/10.1016/j.cageo.2009.05.001>.

[192] SANDERSON, A. R. et al. A framework for exploring numerical solutions of advection-reaction- diffusion equations using a GPU-based approach. *Computing and Visualization in Science*, v. 12, n. 4, p. 155–170, 2009. ISSN 14329360.

[193] CHILDS, H. et al. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In: *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*. [S.l.: s.n.], 2012. p. 357–372.

[194] TUTORIAL do KML - Keyhole Markup Language - Google Developers. Disponível em: <https://developers.google.com/kml/documentation/kml_tut?hl=pt-br>.

[195] SANDERS, J.; KANDROT, E. *CUDA by Example*. [S.l.: s.n.], 2010. v. 54. 279 p. ISSN 1873734X. ISBN 9780131387683.

[196] TARTAKOVSKY, D.; STERN, E.; BRODAY, D. M. Dispersion of tsp and pm10 emissions from quarries in complex terrain. *Science of The Total Environment*, v. 542, n. Part A, p. 946 – 954, 2016. ISSN 0048-9697. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0048969715309499>.

[197] KIM, K.-H.; KABIR, E.; KABIR, S. A review on the human health impact of airborne particulate matter. *Environment International*, v. 74, n. Supplement C, p. 136 – 143, 2015. ISSN 0160-4120. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0160412014002992>.

[198] SAIRANEN, M.; RINNE, M.; SELONEN, O. A review of dust emission dispersions in rock aggregate and natural stone quarries. *International Journal of Mining, Reclamation and Environment*, Taylor and Francis, v. 0, n. 0, p. 1–25, 2017. Disponível em: <https://doi.org/10.1080/17480930.2016.1271385>.

[199] SHUTTLE Radar Topography Mission (SRTM) Void Filled - The Long Term Archive. Disponível em: <https://lta.cr.usgs.gov/SRTMVF>.

[200] SAIRANEN, M.; RINNE, M.; SELONEN, O. A review of dust emission dispersions in rock aggregate and natural stone quarries. *International Journal of Mining, Reclamation and Environment*, Taylor and Francis, v. 0, n. 0, p. 1–25, 2017. Disponível em: <https://doi.org/10.1080/17480930.2016.1271385>.

[201] DEUTSCH, F. et al. Application and validation of a comprehensive model for pm10 and pm2.5 concentrations in belgium and europe. *Applied Mathematical Modelling*, v. 32, n. 8, p. 1501 – 1510, 2008. ISSN 0307-904X. Special issue on numerical and computational issues related to applied mathematical modelling. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0307904X07001576>.

[202] JUSTICE, C. O. et al. The Moderate Resolution Imaging Spectroradiometer (MODIS): Land Remote Sensing for Global Change Research. *IEEE Transactions on Geoscience and Remote Sensing*, v. 36, n. 4, p. 1228–1249, 1998.

[203] SOLANO, R. et al. MODIS Vegetation Index User ' s Guide (MOD13 Series). *The University of Arizona*, v. 2010, n. May, p. 38, 2010.

Appendix A

Advection-reaction-diffusion model code

```
// ******************************************************************
//                       ParMOD2D
// ******************************************************************
// Developer: Tomas Carlotto
// Contact address: thomas.carl@hotmail.com
//*******************************************************************
// Under no circumstances will authors or copyright holders be liable for any claims,
// damages or other liability arising from the use of this model or any part of related code.
//*******************************************************************

// ******************************************************************
//             Description of model inputs and outputs
// ******************************************************************
// INPUTS:
// 1- Conditions:
// 1.1   CT: boundary conditions (matrix format) where:
//           CT = 0 represents inactive and impermeable boundary regions(Neumann condition),
//           CT = 1 represents regions active for the model,
//           CT = -1 represents constant charge regions(Dirichlet condition)
// 1.2   initial_condition: initial conditions matrix
//
// 2- Diffusion Coefficients (Hydraulic Conductivities) and Specific Storage:
// 2.1   DFx: Diffusion coefficient in the x-direction
// 2.2   DFy: Diffusion coefficient in the y-direction
// 2.3   Sy: Specific storage coefficient
//  (For heterogeneous media DFx, DFy and Sy are 2D arrays or line oriented arrays)
//
// 3-  Defining Process Parameters
// 3.1   diffusion: Diffusion process activation parameter.
//           When diffusion = 1, diffusion process is enabled
//           When diffusion = 0, diffusion process is disabled
// 3.2   advection: Advection process activation parameter.
//           When advection = 1, advection process is enabled
//           When advection = 0, advection process is disabled
// 3.3   def_med: Defines whether the medium is homogeneous or heterogeneous
//           When def_med = 0, the medium is homogeneous
//           When def_med = 1, the medium is heterogeneous and arithmetic mean is used in the calculation
//           When def_med = 2, the medium is heterogeneous and harmonic mean is used in the calculation
//           When def_med = 3, the medium is heterogeneous and geometric mean is used in the calculation
// 3.4   timeSimu: Definition of number of time iterations
// 3.5   dt: Sets the time step
// 3.6   resolution: Sets spatial resolution
// 3.7   cols: Number of columns
// 3.8   rows: Number of rows
// 3.9   Cdr: Conductance at the boundary between underground and surface
// 3.10  decay: defines whether the process involves decay (for the dispersion process)
//           When decay = 0, there is no decay
//
// 4  Definition of advection parameters
// 4.1   Vx: Sets the velocity in the x direction
// 4.2   Vy: Sets the velocity in the y direction
// 4.3   mob: Defines the mobility of the solute by the porous medium
//     (In this version of the code it is considered a uniform field of speeds.
//      We will shortly release the non-uniform speed version)
//
// 5 Definition of logistic term parameter
// 5.1 logistic_growth: defines whether the problem involves logistic growth
//     logistic_growth = 1 indicates that there is logistic growth
//     logistic_growth = 0 indicates that there is no logistic growth
// 5.2 ru: Taxa de crescimento logístico
// 5.3 ku: Capacidade de suporte

// OUTPUTS:

// 1 Vector containing the processing times for each time step (TempSimu.txt)
// 2 Vector of strings containing the name of each linear system solution file over time (name_layers.txt)
// 3 File containing linear system dimensions (linear_system_dimension.txt)
// 4 Vector of speeds in the direction x in text file format (Vx.txt)
// 5 Vector of speeds in the y direction in text file format (Vy.txt)
// 6 Vector of solutions of linear systems in text format (U_solution.txt)
// 7 Vector of velocities in the x direction in .mtx format (Vx.mtx)
// 8 Vector of speeds in direction y in .mtx format (Vy.mtx)
// 9 Sequence of solutions for each time step in .mtx format (Resp__ time.mtx)
```

```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"
#include <stdio.h>
#include <cuda.h>
#include <iostream>
#include <stdlib.h>
#include <ctime>
#include <cstdio>
#include <string>
#include <sstream>
#include <math.h>
#include <device_atomic_functions.h>
// cusp include:
//**************
// Input/Output:
//**************
#include <cusp/io/matrix_market.h>
#include <cusp/monitor.h>
//******************
// Matrix containers
//******************
#include <cusp/csr_matrix.h>
#include <cusp/coo_matrix.h>
#include <cusp/hyb_matrix.h>
#include <cusp/dia_matrix.h>
#include <cusp/array2d.h>
#include <cusp/array1d.h>
//********************
// Mathematical functions
//********************
#include <cusp/multiply.h>//
#include <cusp/linear_operator.h>
#include <cusp/elementwise.h>
#include <cusp/convert.h>
// Print
#include <cusp/print.h>
//****************************************************
//  Linear System Solvers (Krylov)
//****************************************************
#include <cusp/krylov/bicgstab.h> // Stabilized Biconjugado Gradient
//************************************
#include <cusp/copy.h>
//**************************************************
//  Kernel of creation of linear system matrices
//**************************************************
#include <gpu_matrix_df.h>
//******************
//  Logistic term for the logistic growth model
//******************
#include <logistic_term.h>
//******************
//**********************************************************************
// Initialization of variables and counters for the rescue functions
std::string tempo;
std::string iterat;
std::string ittime;
std::stringstream out;
std::stringstream oud;
using namespace std;
unsigned int t;
//**********************************************************************
//**************************************************
// Renaming cusp memory allocation functions
//**************************************************
// where to perform the computation
typedef cusp::device_memory MemorySpace;
typedef cusp::host_memory MemorySpaceh;
//************************************
// Definition of variable types
//************************************
typedef int IndexType;
typedef double ValueType;
//obs: To change this value type you must also change the function Logistic_term.h

//*****************************
//Vector subtraction kernel
//*****************************
__global__ void subtract_vet(float *Vet1, float *Vet2, int N, float *subt){
   int tid = blockDim.x*blockIdx.x + threadIdx.x;
   while (tid < N){
      subt[tid] = Vet1[tid] - Vet2[tid];
      tid += gridDim.x * blockDim.x;
   }
}
//*******************************************
// Quadratic error determination kernel
//*******************************************
__global__ void SquareError(int *CT, int *Loc, float *SE, float *Umod, float *Uobs, int N){
   int thid = blockDim.x*blockIdx.x + threadIdx.x;
   while (thid < N){
      if (CT[thid]!=0){
         SE[Loc[thid]] = ((Umod[Loc[thid]] - Uobs[Loc[thid]])*(Umod[Loc[thid]] - Uobs[Loc[thid]]));
      }
      thid += gridDim.x * blockDim.x;
   }
}
// Tree reduction kernel for summation adapted from: "https://wrf.ecse.rpi.edu/wiki/ParallelComputingSpring2015/cuda/git/code-samples-master/posts/parallel_reduction_with_shfl/device_reduce_atomic.h"
//*****************************************************
__global__ void device_reduce_atomic_kernel_vector2(float *in, float* out, int N) {
   float sum = 0;
   int idx = blockIdx.x*blockDim.x + threadIdx.x;
   for (int i = idx; i<N / 2; i += blockDim.x*gridDim.x) {
      float2 val = reinterpret_cast<float2*>(in)[i];
      sum += val.x + val.y;
   }
   int i = idx + N / 2 * 2;
   if (i < N)
   sum += in[i];
   atomicAdd(out, sum);
}
//*****************************************************************************************************
// Kernel for calculating transmissivity or adequacy of diffusivity (in the absence of non-linearity)
//*****************************************************************************************************
__global__ void par_const_TxTy(int storage, int def_med, int *Loc, float *DFx, float *DFy, float depth, float *d_Sy, int
*CT, float *Tx, float *Ty, float *d_lambda, int N){
   int tid2 = blockDim.x*blockIdx.x + threadIdx.x;
   while (tid2 < N){
      if (CT[tid2] != 0){
         if (def_med != 0){
            if (storage == 0){
               d_lambda[Loc[tid2]] = 1 / (2 * d_Sy[tid2]);
            }
            else{
               d_lambda[Loc[tid2]] = 1 / (2 * d_Sy[tid2] * depth);
            }
            Tx[Loc[tid2]] = DFx[tid2] * depth;
            Ty[Loc[tid2]] = DFy[tid2] * depth;
         }
         else {
            if (storage == 0){
               d_lambda[0] = 1 / (2 * d_Sy[0]);
            }
            else{
               d_lambda[0] = 1 / (2 * d_Sy[0] * depth);
            }
            Tx[Loc[tid2]] = DFx[0] * depth;
            Ty[Loc[tid2]] = DFy[0] * depth;
         }
      }
      tid2 += gridDim.x * blockDim.x;
   }
}
//*****************************************************************************************************
// Kernel for calculating transmissivity or adequacy of diffusivity (in the presence of non-linearity)
//*****************************************************************************************************
__global__ void par_var_TxTy(int storage, int def_med, int *Loc, int cols, int rows, float *sup_aquif, float
*upper_limit, float *lower_limit, float *DFx, float *DFy, float *d_Sy, int *CT, float *Tx, float *Ty, float *d_lambda,
int N){
   int tid2 = blockDim.x*blockIdx.x + threadIdx.x;
   while (tid2 < N){
      if (CT[tid2] != 0){
         if (def_med != 0){
            if (storage == 0){
               d_lambda[Loc[tid2]] = 1 / (2 * d_Sy[tid2]);
            }
            else{
               d_lambda[Loc[tid2]] = 1 / (2 * d_Sy[tid2] * (upper_limit[tid2] - lower_limit[tid2]));
            }
            if ((upper_limit[tid2] - sup_aquif[tid2]) <= 0){
               Tx[Loc[tid2]] = DFx[tid2] * (upper_limit[tid2] - lower_limit[tid2]);
               Ty[Loc[tid2]] = DFy[tid2] * (upper_limit[tid2] - lower_limit[tid2]);
            }
            if (((sup_aquif[tid2] - lower_limit[tid2]) > 0) && ((upper_limit[tid2] - sup_aquif[tid2]) > 0)){
               Tx[Loc[tid2]] = DFx[tid2] * (sup_aquif[tid2] - lower_limit[tid2]);
               Ty[Loc[tid2]] = DFy[tid2] * (sup_aquif[tid2] - lower_limit[tid2]);
            }
            if ((sup_aquif[tid2] - lower_limit[tid2]) <= 0){
               if (def_med != 2){
                  Tx[Loc[tid2]] = 0;
                  Ty[Loc[tid2]] = 0;
               }
               else{
                  Tx[Loc[tid2]] = DFx[tid2] *  1e-20;
                  Ty[Loc[tid2]] = DFy[tid2] *  1e-20;
               }
            }
         }
         else {
            if (storage == 0){
               d_lambda[0] = 1 / (2 * d_Sy[0]);
            }
            else{
               d_lambda[0] = 1 / (2 * d_Sy[0] * (upper_limit[tid2] - lower_limit[tid2]));
            }
            if ((upper_limit[tid2] - sup_aquif[tid2]) <= 0){
               Tx[Loc[tid2]] = DFx[0] * (upper_limit[tid2] - lower_limit[tid2]);
               Ty[Loc[tid2]] = DFy[0] * (upper_limit[tid2] - lower_limit[tid2]);
            }
            if (((sup_aquif[tid2] - lower_limit[tid2]) > 0) && ((upper_limit[tid2] - sup_aquif[tid2]) > 0)){
               Tx[Loc[tid2]] = DFx[0] * (sup_aquif[tid2] - lower_limit[tid2]);
               Ty[Loc[tid2]] = DFy[0] * (sup_aquif[tid2] - lower_limit[tid2]);
            }
            if ((sup_aquif[tid2] - lower_limit[tid2]) <= 0){
               if (def_med != 2){
                  Tx[Loc[tid2]] = 0;
                  Ty[Loc[tid2]] = 0;
               }
               else{
                  Tx[Loc[tid2]] = DFx[0] * 1e-20;
                  Ty[Loc[tid2]] = DFy[0] * 1e-20;
               }
            }
         }
      }
      tid2 += gridDim.x * blockDim.x;
   }
}
//*********************************************************************************
// Kernel for the calculation of underground water exits (Base flow)
//*********************************************************************************
__global__ void gw_source_baseflow(int def_med, int *Loc, float *UpperLimitMinus_U, float *dx, float *dy,
float *Sy, int *CT, float *CdrV, float *W, float *Qbase, float Cdr, int N, float dt){
   int in = blockDim.x*blockIdx.x + threadIdx.x;
   while (in < N){
      if (CT[in] != 0){
         if (UpperLimitMinus_U[in] <= 0){
            CdrV[Loc[in]] = Cdr;
            W[Loc[in]] = CdrV[Loc[in]] * UpperLimitMinus_U[in];
            // Flows from the underground system in the form of base flow
            if (def_med != 0){
               Qbase[Loc[in]] = abs(W[Loc[in]] * dx[0] * dy[0])*Sy[in] / dt;
            }
            else{
               Qbase[Loc[in]] = abs(W[Loc[in]] * dx[0] * dy[0])*Sy[0] / dt;
            }
            //*************************************************************
         }
         else {
            W[Loc[in]] = 0;
            CdrV[Loc[in]] = 0;
            Qbase[Loc[in]] = 0;
         }
      }
      in += gridDim.x * blockDim.x;
   }
}

//*****************************************************************************************************
// Kernel for calculating groundwater flow velocities
// Obs: Ky is positive indicating that the origin of the reference system is in the upper left

//    __|______
//      |1 2 3
//      |4 5 6
//      |7 8 9

//*******************************************************************************************************
__global__ void gw_velocity(int def_med, int *CT, int cols, int rows, int *Loc, float *H, float *dy, float *dx, float
*Kx, float *Ky, float *effective_porosity, float *Vx, float *Vy, int N){
   int mov1 = blockDim.x*blockIdx.x + threadIdx.x;
   while (mov1 < N){
      int x = mov1 % cols;
      int y = mov1 / cols;
      if (CT[mov1] != 0){
         if (def_med != 0){
            // Vy
            if ((y > 0) && (y < rows - 1)){
               if ((CT[mov1 - cols] != 0) && (CT[mov1 + cols] != 0) ){
                  Vy[Loc[mov1]] = Ky[mov1] * (H[mov1 + cols] - H[mov1 - cols]) / (2 * dy[0] *
                  effective_porosity[mov1]);
               }
               else if ((CT[mov1 - cols] == 0) && (CT[mov1 + cols] != 0)){
                  Vy[Loc[mov1]] = Ky[mov1] * (H[mov1 + cols] - H[mov1]) / (dy[0] * effective_porosity[mov1]);
               }
               else if ((CT[mov1 + cols] == 0) && (CT[mov1 - cols] != 0)){
                  Vy[Loc[mov1]] = Ky[mov1] * (H[mov1] - H[mov1 - cols]) / (dy[0] * effective_porosity[mov1]);
               }
            }
            if (y == 0){
               Vy[Loc[mov1]] = Ky[mov1] * (H[mov1 + cols] - H[mov1]) / (dy[0] * effective_porosity[mov1]);
            }
            if (y == rows - 1){
               Vy[Loc[mov1]] = Ky[mov1] * (H[mov1] - H[mov1 - cols]) / (dy[0] * effective_porosity[mov1]);
            }
            //Vx
            if ((x > 0) && (x < cols - 1)){
               if ((CT[mov1 + 1] != 0) && (CT[mov1 - 1] != 0) ){
                  Vx[Loc[mov1]] = -Kx[mov1] * (H[mov1 + 1] - H[mov1 - 1]) / (2 * dx[0] * effective_porosity[mov1]);
               }
               else if ((CT[mov1 - 1] == 0) && (CT[mov1 + 1] != 0)){
                  Vx[Loc[mov1]] = -Kx[mov1] * (H[mov1 + 1] - H[mov1]) / (dx[0] * effective_porosity[mov1]);
               }
               else if ((CT[mov1 + 1] == 0) && (CT[mov1 - 1] != 0)){
                  Vx[Loc[mov1]] = -Kx[mov1] * (H[mov1] - H[mov1 - 1]) / (dx[0] * effective_porosity[mov1]);
               }
            }
            if (x == 0){
               Vx[Loc[mov1]] = -Kx[mov1] * (H[mov1 + 1] - H[mov1]) / (dx[0] * effective_porosity[mov1]);
            }
            if (x == cols - 1){
               Vx[Loc[mov1]] = -Kx[mov1] * (H[mov1] - H[mov1 - 1]) / (dx[0] * effective_porosity[mov1]);
            }
         }
         else{
            // Vy
            if ((y > 0) && (y < rows - 1)){
               if ((CT[mov1 - cols] != 0) && (CT[mov1 + cols] != 0) ){
                  Vy[Loc[mov1]] = Ky[0] * (H[mov1 + cols] - H[mov1 - cols]) / (2 * dy[0] * effective_porosity[0]);
               }
               else if ((CT[mov1 - cols] == 0) && (CT[mov1 + cols] != 0)){
                  Vy[Loc[mov1]] = Ky[0] * (H[mov1 + cols] - H[mov1]) / (dy[0] * effective_porosity[0]);
               }
               else if ((CT[mov1 + cols] == 0) && (CT[mov1 - cols] != 0)){
                  Vy[Loc[mov1]] = Ky[0] * (H[mov1] - H[mov1 - cols]) / (dy[0] * effective_porosity[0]);
               }
            }
            if (y == 0){
               Vy[Loc[mov1]] = Ky[0] * (H[mov1 + cols] - H[mov1]) / (dy[0] * effective_porosity[0]);
            }
            if (y == rows - 1){
               Vy[Loc[mov1]] = Ky[0] * (H[mov1] - H[mov1 - cols]) / (dy[0] * effective_porosity[0]);
            }
            //Vx
            if ((x > 0) && (x < cols - 1)){
               if ((CT[mov1 + 1] != 0) && (CT[mov1 - 1] != 0) ){
                  Vx[Loc[mov1]] = -Kx[0] * (H[mov1 + 1] - H[mov1 - 1]) / (2 * dx[0] * effective_porosity[0]);
               }
               else if ((CT[mov1 - 1] == 0) && (CT[mov1 + 1] != 0)){
                  Vx[Loc[mov1]] = -Kx[0] * (H[mov1 + 1] - H[mov1]) / (dx[0] * effective_porosity[0]);
               }
               else if ((CT[mov1 + 1] == 0) && (CT[mov1 - 1] != 0)){
                  Vx[Loc[mov1]] = -Kx[0] * (H[mov1] - H[mov1 - 1]) / (dx[0] * effective_porosity[0]);
               }
            }
            if (x == 0){
               Vx[Loc[mov1]] = -Kx[0] * (H[mov1 + 1] - H[mov1]) / (dx[0] * effective_porosity[0]);
            }
            if (x == cols - 1){
               Vx[Loc[mov1]] = -Kx[0] * (H[mov1] - H[mov1 - 1]) / (dx[0] * effective_porosity[0]);
            }
         }
         //*********************************************
      }
      mov1 += gridDim.x * blockDim.x;
   }
}
//**************************************************************************
// Device function to perform the calculations necessary to generate the linear system corresponding to different
phenomena
//**************************************************************************
__device__ void diff_node(int def_med, int velocity_vec, float *lambda, float *DF, int loc_val, int POS_DF1, int
POS_DF2, float *V, float mob, float *dh, int diffusion, int advection, float *FD_1, float *FD_2, int cadv){
   if ((diffusion == 1) && (advection == 0)){
      if (def_med == 0){
         FD_1[loc_val] = -lambda[0] * (DF[0] / (dh[0] * dh[0]));
         FD_2[loc_val] = lambda[0] * (DF[0] / (dh[0] * dh[0]));
      }
      else if (def_med == 1){
         FD_1[loc_val] = -lambda[POS_DF1] * ((DF[POS_DF1] * dh[0] + DF[POS_DF2] * dh[0]) / (2 * dh[0] * dh[0])) / dh[0];
         FD_2[loc_val] = lambda[POS_DF1] * ((DF[POS_DF1] * dh[0] + DF[POS_DF2] * dh[0]) / (2 * dh[0] * dh[0])) / dh[0];
      }
      else if (def_med == 2){
         FD_1[loc_val] = -lambda[POS_DF1] * ((2 * DF[POS_DF1] * DF[POS_DF2]) / (DF[POS_DF1] * dh[0] + DF[POS_DF2] *
         dh[0])) / dh[0];
         FD_2[loc_val] = lambda[POS_DF1] * ((2 * DF[POS_DF1] * DF[POS_DF2]) / (DF[POS_DF1] * dh[0] + DF[POS_DF2] *
         dh[0])) / dh[0];
      }
      else if (def_med == 3){
         FD_1[loc_val] = -lambda[POS_DF1] * (sqrt(DF[POS_DF1] * DF[POS_DF2]) / sqrt(dh[0] * dh[0])) / dh[0];
         FD_2[loc_val] = lambda[POS_DF1] * (sqrt(DF[POS_DF1] * DF[POS_DF2]) / sqrt(dh[0] * dh[0])) / dh[0];
      }
   }
   if (velocity_vec == 1){
      if ((advection == 1) && (diffusion == 0)){
         if (def_med == 0){
            FD_1[loc_val] = -lambda[0] * (cadv*V[POS_DF2] * mob / (2 * dh[0]));
            FD_2[loc_val] = lambda[0] * (cadv*V[POS_DF2] * mob / (2 * dh[0]));
         }
         else{
            FD_1[loc_val] = -lambda[POS_DF1] * (cadv*V[POS_DF2] * mob / (2 * dh[0]));
            FD_2[loc_val] = lambda[POS_DF1] * (cadv*V[POS_DF2] * mob / (2 * dh[0]));
         }
      }
      if ((diffusion == 1) && (advection == 1)){
         if (def_med == 0){
            FD_1[loc_val] = -lambda[0] * (DF[0] / (dh[0] * dh[0]) + (cadv*V[POS_DF2] * mob / (2 * dh[0])));
            FD_2[loc_val] = lambda[0] * (DF[0] / (dh[0] * dh[0]) + (cadv*V[POS_DF2] * mob / (2 * dh[0])));
         }
         else if (def_med == 1){
            FD_1[loc_val] = -lambda[POS_DF1] * ((((DF[POS_DF1] * dh[0] + DF[POS_DF2] * dh[0]) / (2 * dh[0] * dh[0])) /
            dh[0]) + (cadv*V[POS_DF2] * mob / (2 * dh[0])));
            FD_2[loc_val] = lambda[POS_DF1] * ((((DF[POS_DF1] * dh[0] + DF[POS_DF2] * dh[0]) / (2 * dh[0] * dh[0])) /
            dh[0]) + (cadv*V[POS_DF2] * mob / (2 * dh[0])));
         }
         else if (def_med == 2){
            FD_1[loc_val] = -lambda[POS_DF1] * ((((2 * DF[POS_DF1] * DF[POS_DF2]) / (DF[POS_DF1] * dh[0] + DF[POS_DF2]
            * dh[0])) / dh[0]) + (cadv*V[POS_DF2] * mob / (2 * dh[0])));
            FD_2[loc_val] = lambda[POS_DF1] * ((((2 * DF[POS_DF1] * DF[POS_DF2]) / (DF[POS_DF1] * dh[0] + DF[POS_DF2] *
            dh[0])) / dh[0]) + (cadv*V[POS_DF2] * mob / (2 * dh[0])));
         }
         else if (def_med == 3){
            FD_1[loc_val] = -lambda[POS_DF1] * (((sqrt(DF[POS_DF1] * DF[POS_DF2]) / sqrt(dh[0] * dh[0])) / dh[0]) +
            (cadv*V[POS_DF2] * mob / (2 * dh[0])));
            FD_2[loc_val] = lambda[POS_DF1] * (((sqrt(DF[POS_DF1] * DF[POS_DF2]) / sqrt(dh[0] * dh[0])) / dh[0]) +
            (cadv*V[POS_DF2] * mob / (2 * dh[0])));
         }
      }
   }
   else{
      if ((advection == 1) && (diffusion == 0)){
         if (def_med == 0){
            FD_1[loc_val] = -lambda[0] * (cadv*V[0] * mob / (2 * dh[0]));
            FD_2[loc_val] = lambda[0] * (cadv*V[0] * mob / (2 * dh[0]));
         }
```

```
            else{
               FD_1[loc_val] = -lambda[POS_DF1] * (cadv*V[0] * mob / (2 * dh[0]));
               FD_2[loc_val] = lambda[POS_DF1] * (cadv*V[0] * mob / (2 * dh[0]));
            }
         }
         if ((diffusion == 1) && (advection == 1)){
            if (def_med == 0){
               FD_1[loc_val] = -lambda[0] * (DF[0] / (dh[0] * dh[0]) + (cadv*V[0] * mob / (2 * dh[0])));
               FD_2[loc_val] = lambda[0] * (DF[0] / (dh[0] * dh[0]) + (cadv*V[0] * mob / (2 * dh[0])));
            }
            else if (def_med == 1){
               FD_1[loc_val] = -lambda[POS_DF1] * ((((DF[POS_DF1] * dh[0] + DF[POS_DF2] * dh[0]) / (2 * dh[0] * dh[0])) /
               dh[0]) + (cadv*V[0] * mob / (2 * dh[0])));
               FD_2[loc_val] = lambda[POS_DF1] * ((((DF[POS_DF1] * dh[0] + DF[POS_DF2] * dh[0]) / (2 * dh[0] * dh[0])) /
               dh[0]) + (cadv*V[0] * mob / (2 * dh[0])));
            }
            else if (def_med == 2){
               FD_1[loc_val] = -lambda[POS_DF1] * ((((2 * DF[POS_DF1] * DF[POS_DF2]) / (DF[POS_DF1] * dh[0] + DF[POS_DF2]
               * dh[0])) / dh[0]) + (cadv*V[0] * mob / (2 * dh[0])));
               FD_2[loc_val] = lambda[POS_DF1] * ((((2 * DF[POS_DF1] * DF[POS_DF2]) / (DF[POS_DF1] * dh[0] + DF[POS_DF2] *
               dh[0])) / dh[0]) + (cadv*V[0] * mob / (2 * dh[0])));
            }
            else if (def_med == 3){
               FD_1[loc_val] = -lambda[POS_DF1] * (((sqrt(DF[POS_DF1] * DF[POS_DF2]) / sqrt(dh[0] * dh[0])) / dh[0]) +
               (cadv*V[0] * mob / (2 * dh[0])));
               FD_2[loc_val] = lambda[POS_DF1] * (((sqrt(DF[POS_DF1] * DF[POS_DF2]) / sqrt(dh[0] * dh[0])) / dh[0]) +
               (cadv*V[0] * mob / (2 * dh[0])));
            }
         }
      }
}
// ********************************************************************
// Device function for calculations associated with the central node
// ********************************************************************
__device__ void center(float *lambda, float *FD_1, float *FD_2, int loc_val, int POS_B, int POS_M, int POS_F, int POS_D,
float decay, float dt){
FD_1[loc_val] = (1 / dt) - (FD_1[POS_B] + FD_1[POS_M] + FD_1[POS_F] + FD_1[POS_D] - lambda[POS_B]*decay);//E1[0];
FD_2[loc_val] = (1 / dt) - (FD_2[POS_B] + FD_2[POS_M] + FD_2[POS_F] + FD_2[POS_D] + lambda[POS_B]*decay);//E2[0];
}
// ********************************************************************
// Kernel for assembly of linear system vectors
// ********************************************************************
__global__ void ClassMatPOS(int def_med, int velocity_vec, int diffusion, int advection, float *lambda, float decay,
float mob, int *CT, int *Loc, int *IDrow, int *IDcolumn, float *FD_1, float *FD_2, int cols, int rows, int N,int Nval,
float *DFx, float *DFy, float *Vx, float *Vy, float *dx, float *dy, float dt){
   //Nval is the number of active elements
   int mov = blockDim.x*blockIdx.x + threadIdx.x;
   while (mov < N){
      int x = mov % cols;
      int y = mov / cols;
      if (CT[mov] != 0){
         // TOP
         if (y > 0){
            if (CT[mov - cols] == 0){
               IDcolumn[Loc[mov]] = -1;
               IDrow[Loc[mov]] = Loc[mov];
            }
            else{
               IDcolumn[Loc[mov]] =  Loc[mov - cols];
               IDrow[Loc[mov]] = Loc[mov];
               diff_node(def_med,velocity_vec, lambda, DFy, Loc[mov], Loc[mov], Loc[mov - cols], Vy, mob, dy,
               diffusion, advection, FD_1, FD_2, -1); //-Vy
            }
         }
         // BOTTOM
         //__syncthreads();
         if (y < rows - 1){
            if (CT[mov + cols] == 0){
               IDcolumn[Loc[mov] + Nval] = -1;
               IDrow[Loc[mov] + Nval] = Loc[mov];
            }
            else{
               IDcolumn[Loc[mov] + Nval] = Loc[mov + cols];
               IDrow[Loc[mov] + Nval] = Loc[mov];
               diff_node(def_med, velocity_vec, lambda, DFy, Loc[mov] + Nval, Loc[mov], Loc[mov + cols], Vy, mob, dy,
               diffusion, advection, FD_1, FD_2, 1);
            }
         }
         //RIGHT
         //__syncthreads();
         if (x < cols - 1){
            if (CT[mov + 1] == 0){
               IDcolumn[Loc[mov] + 2 * Nval] = -1;
               IDrow[Loc[mov] + 2 * Nval] = Loc[mov];
            }
            else{
               IDcolumn[Loc[mov] + 2 * Nval] = Loc[mov] + 1;
               IDrow[Loc[mov] + 2 * Nval] = Loc[mov];
               diff_node(def_med, velocity_vec, lambda, DFx, Loc[mov] + 2 * Nval, Loc[mov], Loc[mov] + 1, Vx, mob, dx,
               diffusion, advection, FD_1, FD_2, -1);
            }
         }
         // LEFT
         //__syncthreads();
         if (x > 0){
            if (CT[mov - 1] == 0){
               IDcolumn[Loc[mov] + 3 * Nval] = -1;
               IDrow[Loc[mov] + 3 * Nval] = Loc[mov];
            }
            else{
               IDcolumn[Loc[mov] + 3 * Nval] = Loc[mov] - 1;
               IDrow[Loc[mov] + 3 * Nval] = Loc[mov];
               diff_node(def_med, velocity_vec, lambda, DFx, Loc[mov] + 3 * Nval, Loc[mov], Loc[mov] - 1, Vx, mob, dx,
               diffusion, advection, FD_1, FD_2, 1);
            }
         }
         //__syncthreads();
         //========================= Internal Diagonals =====================
         if ((IDcolumn[Loc[mov] + 2 * Nval] == -1) && (IDcolumn[Loc[mov] + 3 * Nval] != -1)){
            FD_1[Loc[mov] + 3 * Nval] =  2 * FD_1[Loc[mov] + 3 * Nval]; //D
            FD_2[Loc[mov] + 3 * Nval] =  2 * FD_2[Loc[mov] + 3 * Nval]; //D
         }
         if ((IDcolumn[Loc[mov] + 2 * Nval] != -1) && (IDcolumn[Loc[mov] + 3 * Nval] == -1)){
            FD_1[Loc[mov] + 2 * Nval] =  2 * FD_1[Loc[mov] + 2 * Nval]; //F
            FD_2[Loc[mov] + 2 * Nval] =  2 * FD_2[Loc[mov] + 2 * Nval]; //F
         }
         //========================= External diagonals ====================
         if ((IDcolumn[Loc[mov]] == -1) && (IDcolumn[Loc[mov] + Nval] != -1)){
            FD_1[Loc[mov] + Nval] =  2 * FD_1[Loc[mov] + Nval]; //M
            FD_2[Loc[mov] + Nval] =  2 * FD_2[Loc[mov] + Nval]; //M
         }
         if ((IDcolumn[Loc[mov]] != -1) && (IDcolumn[Loc[mov] + Nval] == -1)){
            FD_1[Loc[mov]] =  2 * FD_1[Loc[mov]]; //B
            FD_2[Loc[mov]] =  2 * FD_2[Loc[mov]]; //B
         }
         //=================================================================
         //========================  Diagonal Principal =====================
         IDrow[Loc[mov] + 4 * Nval] = Loc[mov];
         IDcolumn[Loc[mov] + 4 * Nval] = Loc[mov];
         center(lambda, FD_1, FD_2, Loc[mov] + 4 * Nval, Loc[mov], Loc[mov] + Nval, Loc[mov] + 2 * Nval, Loc[mov] + 3 *
         Nval, decay, dt);
      }
      mov += gridDim.x * blockDim.x;
   }
}
//*************************************************************************************
// CUSP function for the calculation of the logistic term
//*************************************************************************************
cusp::array1d<ValueType, MemorySpace> logistic_term(cusp::array1d<ValueType, MemorySpace> Mrub,float
ru,cusp::array1d<int, MemorySpace> un,cusp::array1d<ValueType, MemorySpace> diff_unb,cusp::array1d<ValueType,
MemorySpace> L,cusp::array1d<ValueType, MemorySpace> x){
   //*************************************************************************************
   //L = ru*b(un-b);    // Logistic term
   //*************************************************************************************
   // Multiplication of vector b by scalar value ru: (Mrub = (ru*b))
   //Mrub = x;
   cusp::blas::copy(x, Mrub);
   cusp::blas::scal(Mrub, ru); //Mrub = ru*Mrub;
   // Defining the difference vector between a and b: (diff_unb = (un-b))
   cusp::blas::axpby(un, x, diff_unb, 1, -1);
   // element-by-element multiplication elemento (z[i] = x[i] * y[i])
   cusp::blas::xmy(Mrub, diff_unb, L);
   //*************************************************************************************
   //+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
   return L;
}
```

```
//*********************************
// Main Code
//*********************************

int main(){
   // Start of the time controller
   //std::clock_t start;
   //double duration;
   //start = std::clock();
   //*************************************
   // Initialization of host variables
   //*************************************
   int i, in, N, cols, rows, Nsp, cont, Nsparse, timeSimu, timeSimu1, timeSimu2, velocity_vec, Nvelocity;
   int numBlocks;                     //Number of blocks
   int threadsPerBlock;               //Number of threads
   int maxThreadsPerBlock;
   float *h_initial_condition, *h_upper_limit, *h_lower_limit, *h_Sy,*h_ne, *h_DFx, *h_DFy, *h_Tx, *h_dx, *h_dy,
   *h_x,*h_y, *h_Hconst, *h_W, *h_Qbase, *h_Vx_vec, *h_Vy_vec, *h_Vx, *h_Vy, **U2d;
   int *h_CT, *h_Loc;
   float *h_FD_1, *h_FD_2;
   int *h_IDrow, *h_IDcolumn;
   float ru, decay, mob, Vx, Vy, resolution, dt, Cdr, diff_upper_lower, ku;
   float *h_UpperVar,*h_LowerVar;
   float *d_initial_condition, *d_upper_limit, *d_lower_limit, *d_Sy,*d_ne, *d_DFx, *d_DFy, *d_Tx, *d_Ty, *d_lambda,
   *d_dx,*d_dy, *d_x, *d_y, *d_Hconst, *d_W, *d_CdrV, *d_Qbase, *d_Vx_vec, *d_Vy_vec, *d_Vx, *d_Vy;
   int *d_CT, *d_Loc;
   float *d_FD_1, *d_FD_2;
   int *d_IDrow, *d_IDcolumn;
   float *d_UpperVar, *d_LowerVar;
   int file_mtx, file_txt, save_final_result, save_result_sequence, save_step,
   save_times,off_save_sequence,save_velocity;
   int MemDim, Memlambda, def_med, diffusion, advection, upper_limit_on, lower_limit_on, base_flow, logistic_growth,
   multiSys, dir_number, dir_it,storage;
   std::string dir_parameters, north, south, east, west, outNameLayers, dirNvalOutput, dirFiletype, dirSaveFinalResult,
   dirSaveResultSequence, dirSaveTimes, dirTimeOutput, dirSaveVelocity, dirStorage;
   double duration;
   char dirfile[4000];
   //*************************************************
   FILE *dir = fopen("db\\dir.txt", "r");
   fscanf(dir, " dir_number: %d/n", &dir_number);
   for (dir_it = 0; dir_it < dir_number; dir_it++){
      fscanf(dir, " %s/n", dirfile);
      dir_parameters = dirfile;
      //************************************
      // Data save settings
      //***************************************************************
      // Output file format
      FILE *file_type;
      dirFiletype = "db\\" + dir_parameters + "\\input\\save_settings\\selected_file_type\\file_type.txt";
      file_type = fopen(dirFiletype.c_str(), "r");
      fscanf(file_type, " file_mtx: %d/n", &file_mtx);
      fscanf(file_type, " file_txt: %d/n", &file_txt);
      fclose(file_type);
      // Selecting output files
      FILE *SaveFinalResult;
      dirSaveFinalResult = "db\\" + dir_parameters + "\\input\\save_settings\\selected_output\\output_final_result.txt";
      SaveFinalResult = fopen(dirSaveFinalResult.c_str(), "r");
      fscanf(SaveFinalResult, " save_final_result: %d/n", &save_final_result);
      fclose(SaveFinalResult);
      // Selecting the save sequence
      FILE *SaveResultSequence;
      dirSaveResultSequence = "db\\" + dir_parameters +
      "\\input\\save_settings\\selected_output\\output_result_sequence.txt";
      SaveResultSequence = fopen(dirSaveResultSequence.c_str(), "r");
      fscanf(SaveResultSequence, " save_result_sequence: %d/n", &save_result_sequence);
      fscanf(SaveResultSequence, " save_step: %d/n", &save_step);
      fclose(SaveResultSequence);
      // Time saving selection
      FILE *SaveTimes;
      dirSaveTimes = "db\\" + dir_parameters + "\\input\\save_settings\\selected_output\\output_times.txt";
      SaveTimes = fopen(dirSaveTimes.c_str(), "r");
      fscanf(SaveTimes, " save_times: %d/n", &save_times);
      fclose(SaveTimes);
      // Speed rescue selection
      FILE *SaveVelocity;
      dirSaveVelocity = "db\\" + dir_parameters + "\\input\\save_settings\\selected_output\\output_velocity.txt";
      SaveVelocity = fopen(dirSaveVelocity.c_str(), "r");
      fscanf(SaveVelocity, " save_velocity: %d/n", &save_velocity);
      fclose(SaveVelocity);
      //***************************************************************
      // Save text file containing linear system dimensions
```

```
FILE *Nval_output;
dirNvalOutput = "db\\" + dir_parameters + "\\output\\linear_system_dimension.txt";
Nval_output = fopen(dirNvalOutput.c_str(), "w");
// Save txt file with the names of the layers (responses in each step of time)
FILE *writeNameLayers;
if ((file_txt == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
   outNameLayers = "db\\" + dir_parameters + "\\output\\solution_txt\\name_layers.txt";
   writeNameLayers = fopen(outNameLayers.c_str(), "w");
}
std::string limits_param;
limits_param = "db\\" + dir_parameters + "\\input\\parameters\\limits.txt";
//Definition Upper Limit and Lower Limit
FILE *limitOn = fopen(limits_param.c_str(), "r");
if (limitOn == NULL) {
   printf("unknown file - limits.txt\n");
   system("pause");
   return 0;
}
fscanf(limitOn, " upper_limit_on: %d/n", &upper_limit_on);
fscanf(limitOn, " lower_limit_on: %d/n", &lower_limit_on);
if ((upper_limit_on == 0) && (lower_limit_on == 0)){
   fscanf(limitOn, " diff_upper_lower: %f/n", &diff_upper_lower);
}
fclose(limitOn);
// Storage Type Configuration
FILE *inputStorage;
dirStorage = "db\\" + dir_parameters + "\\input\\parameters\\storage.txt";
inputStorage = fopen(dirStorage.c_str(), "r");
if (inputStorage == NULL) {
   printf("unknown file - storage.txt\n");
   system("pause");
   return 0;
}
fscanf(inputStorage,"storage: %d\n", &storage);
fclose(inputStorage);
// Source term (model of groundwater)
std::string source_base_flow;
source_base_flow = "db\\" + dir_parameters + "\\input\\sources\\base_flow.txt";
FILE *source_bf = fopen(source_base_flow.c_str(), "r");
if (source_bf == NULL) {
   printf("unknown file - base_flow.txt\n");
   system("pause");
   return 0;
}
fscanf(source_bf, " base_flow: %d/n", &base_flow);
if (base_flow == 1){
   fscanf(source_bf, " Cdr: %f/n", &Cdr);
}
fclose(source_bf);
std::string source_logistic_growth;
source_logistic_growth = "db\\" + dir_parameters + "\\input\\sources\\logistic_growth.txt";
FILE *source_lg = fopen(source_logistic_growth.c_str(), "r");
if (source_lg == NULL) {
   printf("unknown file - logistic_growth.txt\n");
   system("pause");
   return 0;
}
fscanf(source_lg, " logistic_growth: %d/n", &logistic_growth);
if (logistic_growth == 1){
   //Logistic growth rate
   fscanf(source_lg, " ru: %f/n", &ru);
   // Support capability
   fscanf(source_lg, " k: %f/n", &ku);
}
fclose(source_lg);
//Parameters of the physical phenomenon
std::string file_param;
file_param = "db\\" + dir_parameters + "\\input\\parameters\\param.txt";
FILE *param = fopen(file_param.c_str(), "r");
if (param == NULL) {
   printf("unknown file - param.txt\n");
   system("pause");
   return 0;
}
fscanf(param, " resolution: %f/n", &resolution);
fscanf(param, " timeSimu: %d/n", &timeSimu);
fscanf(param, " dt: %f/n", &dt);
fscanf(param, " diffusion: %d/n", &diffusion);
fscanf(param, " advection: %d/n", &advection);
fscanf(param, " decay: %f/n", &decay);
// Definition of linear system variability with time
```

```
std::string multiSys_param;
multiSys_param = "db\\" + dir_parameters + "\\input\\parameters\\multiSys.txt";
FILE *param_multiSys = fopen(multiSys_param.c_str(), "r");
if (param == NULL) {
   printf("unknown file - multiSys.txt\n");
   system("pause");
   return 0;
}
fscanf(param_multiSys, " multiSys: %d/n", &multiSys);
// Definition of the medium (homogeneous / heterogeneous)
std::string def_med_param;
def_med_param = "db\\" + dir_parameters + "\\input\\parameters\\def_med.txt";
FILE *param_def_med = fopen(def_med_param.c_str(), "r");
if (param_def_med == NULL) {
   printf("unknown file - def_med.txt\n");
   system("pause");
   return 0;
}
fscanf(param_def_med, " def_med: %d/n", &def_med);
// Parameters of the term advective
std::string velocity_param;
velocity_param = "db\\" + dir_parameters + "\\input\\parameters\\def_velocity.txt";
FILE *param_velocity = fopen(velocity_param.c_str(), "r");
if (param_velocity == NULL) {
   printf("unknown file - def_velocity.txt\n");
   system("pause");
   return 0;
}
fscanf(param_velocity, " velocity_vec: %d\n", &velocity_vec);
// Mobility
std::string file_param_advection;
file_param_advection = "db\\" + dir_parameters + "\\input\\parameters\\param_advection.txt";
FILE *param_adv = fopen(file_param_advection.c_str(), "r");
if (param_adv == NULL) {
   printf("unknown file - param_advection.txt\n");
   system("pause");
   return 0;
}
fscanf(param_adv, " mob: %f\n", &mob);
// Initial and contour conditions and calculation of number of elements (N)
std::string initial_condition, boundary_conditions;
initial_condition = "db\\" + dir_parameters + "\\input\\conditions\\initial_condition.txt";
boundary_conditions = "db\\" + dir_parameters + "\\input\\conditions\\CT.txt";
// Initial condition
FILE *V_initial_condition = fopen(initial_condition.c_str(), "r");
if (V_initial_condition == NULL) {
   printf("unknown file - initial_condition.txt\n");
   system("pause");
   return 0;
}
// Contour Conditions
FILE *V_CT = fopen(boundary_conditions.c_str(), "r");
if (V_CT == NULL) {
   printf("unknown file - CT.txt\n");
   system("pause");
   return 0;
}
// Geographical coordinates of the study area
fscanf(V_initial_condition, " north: %s\n", &north); fscanf(V_CT, " north: %s\n", &north);
fscanf(V_initial_condition, " south: %s\n", &south); fscanf(V_CT, " south: %s\n", &south);
fscanf(V_initial_condition, " east: %s\n", &east); fscanf(V_CT, " east: %s\n", &east);
fscanf(V_initial_condition, " west: %s\n", &west); fscanf(V_CT, " west: %s\n", &west);
fscanf(V_initial_condition, " rows: %d\n", &rows); fscanf(V_CT, " rows: %d\n", &rows);
fscanf(V_initial_condition, " cols: %d\n", &cols); fscanf(V_CT, " cols: %d\n", &cols);
N = cols*rows;
h_initial_condition = (float*)malloc(N*sizeof(float));
h_CT = (int*)malloc(N*sizeof(int));
cudaMalloc((void**)&d_initial_condition, N*sizeof(float));
cudaMalloc((void**)&d_CT, N*sizeof(int));
for (i = 0; i < N; i++){
   fscanf(V_initial_condition, "%f\n", &h_initial_condition[i]);
   fscanf(V_CT, "%d\n", &h_CT[i]);
}
//****************************************************
//Allocate memory in CPU
//****************************************************
U2d = (float**)malloc(rows*sizeof(float*));
for (int iu = 0; iu < rows; iu++){
   U2d[iu] = (float*)malloc(cols*sizeof(float));
}
h_Loc = (int*)malloc(N*sizeof(int));
```

```
h_dx = (float*)malloc(sizeof(float));
h_dy = (float*)malloc(sizeof(float));
h_x = (float*)malloc(cols*sizeof(float));
h_y = (float*)malloc(rows*sizeof(float));
h_Hconst = (float*)malloc(N*sizeof(float));
//****************************************************
// Allocate memory on the GPU
//****************************************************
cudaMalloc((void**)&d_Loc, N*sizeof(int));
cudaMalloc((void**)&d_dx, sizeof(float));
cudaMalloc((void**)&d_dy, sizeof(float));
cudaMalloc((void**)&d_x, cols*sizeof(float));
cudaMalloc((void**)&d_y, rows*sizeof(float));
cudaMalloc((void**)&d_Hconst, N*sizeof(float));
if (upper_limit_on == 1){
   std::string dir_upper_limit;
   h_upper_limit = (float*)malloc(N*sizeof(float));
   cudaMalloc((void**)&d_upper_limit, N*sizeof(float));
   // Allocation of variable h_UpperVar to receive values of operations involving the upper bound
   h_UpperVar = (float*)malloc(N*sizeof(float));
   cudaMalloc((void**)&d_UpperVar, N*sizeof(float));
   dir_upper_limit = "db\\" + dir_parameters + "\\input\\limits\\upper_limit.txt";
   FILE *V_upper_limit = fopen(dir_upper_limit.c_str(), "r");
   if (V_upper_limit == NULL) {
      printf("unknown file - upper_limit.txt\n");
      system("pause");
      return 0;
   }
   fscanf(V_upper_limit, " north: %s\n", &north);
   fscanf(V_upper_limit, " south: %s\n", &south);
   fscanf(V_upper_limit, " east: %s\n", &east);
   fscanf(V_upper_limit, " west: %s\n", &west);
   fscanf(V_upper_limit, " rows: %d\n", &rows);
   fscanf(V_upper_limit, " cols: %d\n", &cols);
   for (i = 0; i < N; i++){
      fscanf(V_upper_limit, "%f\n", &h_upper_limit[i]);
   }
   cudaMemcpy(d_upper_limit, h_upper_limit, N*sizeof(float), cudaMemcpyHostToDevice);
}
if (lower_limit_on == 1){
   std::string dir_lower_limit;
   h_lower_limit = (float*)malloc(N*sizeof(float));
   cudaMalloc((void**)&d_lower_limit, N*sizeof(float));
   // Allocation of variable h_LowerVar to receive values of operations involving the lower bound
   h_LowerVar = (float*)malloc(N*sizeof(float));
   cudaMalloc((void**)&d_LowerVar, N*sizeof(float));
   dir_lower_limit = "db\\" + dir_parameters + "\\input\\limits\\lower_limit.txt";
   FILE *V_lower_limit = fopen(dir_lower_limit.c_str(), "r");
   if (V_lower_limit == NULL) {
      printf("unknown file - lower_limit.txt\n");
      system("pause");
      return 0;
   }
   fscanf(V_lower_limit, " north: %s\n", &north);
   fscanf(V_lower_limit, " south: %s\n", &south);
   fscanf(V_lower_limit, " east: %s\n", &east);
   fscanf(V_lower_limit, " west: %s\n", &west);
   fscanf(V_lower_limit, " rows: %d\n", &rows);
   fscanf(V_lower_limit, " cols: %d\n", &cols);
   for (i = 0; i < N; i++){
      fscanf(V_lower_limit, "%f\n", &h_lower_limit[i]);
   }
   cudaMemcpy(d_lower_limit, h_lower_limit, N*sizeof(float), cudaMemcpyHostToDevice);
}
//***********************************************************
// Properties of medium or material
//***********************************************************
std::string diffusion_DFx, diffusion_DFy, coef_Sy, effective_porosity;
if (def_med != 0){
   diffusion_DFx = "db\\" + dir_parameters + "\\input\\heterogeneous\\DFx.txt";
   diffusion_DFy = "db\\" + dir_parameters + "\\input\\heterogeneous\\DFy.txt";
   coef_Sy = "db\\" + dir_parameters + "\\input\\heterogeneous\\Sy.txt";
   effective_porosity = "db\\" + dir_parameters + "\\input\\heterogeneous\\ne.txt";
   MemDim = N;
}
else{
   diffusion_DFx = "db\\" + dir_parameters + "\\input\\homogeneous\\DFx.txt";
   diffusion_DFy = "db\\" + dir_parameters + "\\input\\homogeneous\\DFy.txt";
   coef_Sy = "db\\" + dir_parameters + "\\input\\homogeneous\\Sy.txt";
   effective_porosity = "db\\" + dir_parameters + "\\input\\homogeneous\\ne.txt";
   MemDim = 1;
}
h_ne = (float*)malloc(MemDim*sizeof(float));
h_Sy = (float*)malloc(MemDim*sizeof(float));
h_DFx = (float*)malloc(MemDim*sizeof(float));
h_DFy = (float*)malloc(MemDim*sizeof(float));
cudaMalloc((void**)&d_ne, MemDim*sizeof(float));
cudaMalloc((void**)&d_Sy, MemDim*sizeof(float));
cudaMalloc((void**)&d_DFx, MemDim*sizeof(float));
cudaMalloc((void**)&d_DFy, MemDim*sizeof(float));
FILE *V_ne;
if (save_velocity == 1){
   V_ne = fopen(effective_porosity.c_str(), "r");
   if (V_ne == NULL) {
      printf("unknown file - ne.txt\n");
      system("pause");
      return 0;
   }
}
FILE *V_Sy = fopen(coef_Sy.c_str(), "r");
if (V_Sy == NULL) {
   printf("unknown file - Sy.txt\n");
   system("pause");
   return 0;
}
FILE *V_DFx = fopen(diffusion_DFx.c_str(), "r");
if (V_DFx == NULL) {
   printf("unknown file - DFx.txt\n");
   system("pause");
   return 0;
}
FILE *V_DFy = fopen(diffusion_DFy.c_str(), "r");
if (V_DFy == NULL) {
   printf("unknown file - DFy.txt\n");
   system("pause");
   return 0;
}
fscanf(V_Sy, " north: %s\n", &north); fscanf(V_DFx, " north: %s\n", &north); fscanf(V_DFy, " north: %s\n",
&north);
fscanf(V_Sy, " south: %s\n", &south); fscanf(V_DFx, " south: %s\n", &south); fscanf(V_DFy, " south: %s\n",
&south);
fscanf(V_Sy, " east: %s\n", &east); fscanf(V_DFx, " east: %s\n", &east); fscanf(V_DFy, " east: %s\n", &east);
fscanf(V_Sy, " west: %s\n", &west); fscanf(V_DFx, " west: %s\n", &west); fscanf(V_DFy, " west: %s\n", &west);
fscanf(V_Sy, " rows: %d\n", &rows); fscanf(V_DFx, " rows: %d\n", &rows); fscanf(V_DFy, " rows: %d\n", &rows);
fscanf(V_Sy, " cols: %d\n", &cols); fscanf(V_DFx, " cols: %d\n", &cols); fscanf(V_DFy, " cols: %d\n", &cols);
if (save_velocity == 1){
   fscanf(V_ne, " north: %s\n", &north);
   fscanf(V_ne, " south: %s\n", &south);
   fscanf(V_ne, " east: %s\n", &east);
   fscanf(V_ne, " west: %s\n", &west);
   fscanf(V_ne, " rows: %d\n", &rows);
   fscanf(V_ne, " cols: %d\n", &cols);
}
for (int mem = 0; mem < MemDim; mem++){
   if (save_velocity==1){
      fscanf(V_ne, " %f\n", &h_ne[mem]);
   }
   fscanf(V_Sy, " %f\n", &h_Sy[mem]);
   fscanf(V_DFx, " %f\n", &h_DFx[mem]);
   fscanf(V_DFy, " %f\n", &h_DFy[mem]);
}
// *******************************************************
// Information displayed at the prompt
//********************************************************
printf("****************************************************\n");
printf("******************* ParMOD2D ********************\n");
printf("****************************************************\n");
printf("north: %s\n", north);
printf("south: %s\n", south);
printf("east: %s\n", east);
printf("west: %s\n", west);
printf("cols: %d\n", cols);
printf("rows: %d\n", rows);
printf("dir_number: %d\n", dir_number);
printf("dir: %s\n", dir_parameters.c_str());
printf("diffusion: %d\n", diffusion);
printf("advection: %d\n", advection);
printf("N: %d\n", N);
printf("resolution: %f\n", resolution);
printf("timeSimu: %d\n", timeSimu);
printf("dt: %f\n", dt);
printf("multiSys: %d\n", multiSys);
printf("def_med: %d\n", def_med);
printf("decay: %f\n", decay);
printf("mob: %f\n", mob);
printf("****************************************************\n");
printf("****************************************************\n");

float x0;
float y0;
unsigned int px;
unsigned int py;

// Mesh definitions
x0 = 0;
y0 = 0;
px = cols;
py = rows;
h_dx[0] = resolution;
h_dy[0] = resolution;
FILE *writeX, *writeY;
writeX = fopen("X.txt", "w");
for (int i = 0; i < px; i++){
   h_x[i] = x0 + i*h_dx[0];
   fprintf(writeX, " %f\n", h_x[i]);
}
fclose(writeX);
writeY = fopen("Y.txt", "w");
for (int j = 0; j < py; j++){
   h_y[j] = y0 + j*h_dy[0];
   fprintf(writeY, " %f\n", h_y[j]);
}
fclose(writeY);
// ********* Mapping and numbering of active positions ***************
int k = -1;
int Nval = 0;
for (int in = 0; in < N; in++) {
   h_Loc[in] = 0;
   if (h_CT[in] != 0){
      Nval++;
      k = k + 1;
      h_Loc[in] = k;
   }
}
cudaMemcpy(d_Loc, h_Loc, N*sizeof(int), cudaMemcpyHostToDevice); // Copy to GPU
printf("Linear System Size: %d lines x %d columns\n", Nval, Nval);
fprintf(Nval_output, " %d\n", Nval);
fclose(Nval_output);
if (def_med != 0){
   Memlambda = Nval;
}
else{
   Memlambda = 1;
}
if (multiSys == 1){
   timeSimu1 = timeSimu;
   timeSimu2 = 1;
}
else{
   timeSimu2 = timeSimu;
   timeSimu1 = 1;
}
int dimSys = Nval * 5;
h_Tx = (float*)malloc(Nval*sizeof(float));
cudaMalloc((void**)&d_Tx, Nval*sizeof(float));
cudaMalloc((void**)&d_Ty, Nval*sizeof(float));
h_Vx_vec = (float*)malloc(Nval*sizeof(float));
h_Vy_vec = (float*)malloc(Nval*sizeof(float));
cudaMalloc((void**)&d_Vx_vec, Nval*sizeof(float));
cudaMalloc((void**)&d_Vy_vec, Nval*sizeof(float));
cudaMalloc((void**)&d_lambda, Memlambda*sizeof(float));
h_W = (float*)malloc(Nval*sizeof(float));
cudaMalloc((void**)&d_W, Nval*sizeof(float));
cudaMalloc((void**)&d_CdrV, Nval*sizeof(float));
h_Qbase = (float*)malloc(Nval*sizeof(float));
cudaMalloc((void**)&d_Qbase, Nval*sizeof(float));
h_IDrow = (int*)malloc(dimSys*sizeof(int));
h_IDcolumn = (int*)malloc(dimSys*sizeof(int));
h_FD_1 = (float*)malloc(dimSys*sizeof(float));
h_FD_2 = (float*)malloc(dimSys*sizeof(float));
cudaMalloc((void**)&d_IDrow, dimSys*sizeof(int));
cudaMalloc((void**)&d_IDcolumn, dimSys*sizeof(int));
cudaMalloc((void**)&d_FD_1, dimSys*sizeof(float));
cudaMalloc((void**)&d_FD_2, dimSys*sizeof(float));
// Allocation of CUSP vectors
cusp::array1d<ValueType, MemorySpace> b(Nval, 0);
cusp::array1d<ValueType, MemorySpace> H(Nval, 0);
cusp::array1d<ValueType, MemorySpace> Wn(Nval, 0);
cusp::array1d<ValueType, MemorySpace> Ws(Nval, 0);
cusp::array2d<ValueType, MemorySpace> Vx_vec(rows, cols);
cusp::array2d<ValueType, MemorySpace> Vy_vec(rows, cols);
//Source term
//Logistic growth
*********************************************************
cusp::array1d<ValueType, MemorySpace> Mrub(Nval, 0);
cusp::array1d<int, MemorySpace> un(Nval, 1);
cusp::array1d<ValueType, MemorySpace> diff_unb(Nval, 0);
// ****************************************************************************
cusp::array1d<ValueType, MemorySpace> testeSubtract(Nval, 0);
// Velocities / advective term
std::string velocity_Vx, velocity_Vy;
if (velocity_vec == 0){
   velocity_Vx = "db\\" + dir_parameters + "\\input\\velocity_const\\Vx.txt";
   velocity_Vy = "db\\" + dir_parameters + "\\input\\velocity_const\\Vy.txt";
   Nvelocity = 1;
}
else{
   velocity_Vx = "db\\" + dir_parameters + "\\input\\velocity_vec\\Vx.txt";
   velocity_Vy = "db\\" + dir_parameters + "\\input\\velocity_vec\\Vy.txt";
   Nvelocity = Nval;
}
h_Vx = (float*)malloc(Nvelocity*sizeof(float));
h_Vy = (float*)malloc(Nvelocity*sizeof(float));
FILE *V_Vx = fopen(velocity_Vx.c_str(), "r");
if (V_Vx == NULL) {
   printf("unknown file - Vx.txt\n");
   system("pause");
   return 0;
}
FILE *V_Vy = fopen(velocity_Vy.c_str(), "r");
if (V_Vy == NULL) {
   printf("unknown file - Vy.txt\n");
   system("pause");
   return 0;
}
fscanf(V_Vx, " north: %s\n", &north); fscanf(V_Vy, " north: %s\n", &north);
fscanf(V_Vx, " south: %s\n", &south); fscanf(V_Vy, " south: %s\n", &south);
fscanf(V_Vx, " east: %s\n", &east); fscanf(V_Vy, " east: %s\n", &east);
fscanf(V_Vx, " west: %s\n", &west); fscanf(V_Vy, " west: %s\n", &west);
fscanf(V_Vx, " rows: %d\n", &rows); fscanf(V_Vy, " rows: %d\n", &rows);
fscanf(V_Vx, " cols: %d\n", &cols); fscanf(V_Vy, " cols: %d\n", &cols);
for (int vel = 0; vel < Nvelocity; vel++){
   fscanf(V_Vx, " %f\n", &h_Vx[vel]);
   fscanf(V_Vy, " %f\n", &h_Vy[vel]);
}
cudaMalloc((void**)&d_Vx, Nvelocity*sizeof(float));
cudaMalloc((void**)&d_Vy, Nvelocity*sizeof(float));
cudaMemcpy(d_Vx, h_Vx, Nvelocity*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_Vy, h_Vy, Nvelocity*sizeof(float), cudaMemcpyHostToDevice);
//float*atlH;
//atlH = (float*)malloc(Nval*sizeof(float));
int kh = -1;
for (in = 0; in < N; in++){
   if (h_CT[in] != 0){
      kh++;
      H[kh] = h_initial_condition[in];
      if (h_CT[in] < 0){
         h_Hconst[kh] = h_initial_condition[in];
      }
   }
}
int n_ativos = 0;
for (int in = 0; in < N; in++) { if (h_CT[in] != 0){ n_ativos++; } }
cusp::coo_matrix<int, ValueType, MemorySpaceh> Umtx(rows, cols, n_ativos);
// ******************************************************************
// Copying the vectors from the CPU to the GPU
// ******************************************************************
cudaMemcpy(d_initial_condition, h_initial_condition, N*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_CT, h_CT, N*sizeof(int), cudaMemcpyHostToDevice);
if (save_velocity == 1){
   cudaMemcpy(d_ne, h_ne, MemDim*sizeof(float), cudaMemcpyHostToDevice);
}
cudaMemcpy(d_Sy, h_Sy, MemDim*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_DFx, h_DFx, MemDim*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_DFy, h_DFy, MemDim*sizeof(float), cudaMemcpyHostToDevice);
//cudaMemcpy(d_dx, h_dx, cols*sizeof(float), cudaMemcpyHostToDevice);
//cudaMemcpy(d_dy, h_dy, rows*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_dx, h_dx, sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_dy, h_dy, sizeof(float), cudaMemcpyHostToDevice);
// *******************************************************************
// Definition of the number of blocks and threads for mesh (N cells)
// *******************************************************************
cudaDeviceProp prop;
int count;
cudaGetDeviceCount(&count);
for (int i = 0; i < count; i++){
   cudaGetDeviceProperties(&prop, i);
   maxThreadsPerBlock = prop.maxThreadsPerBlock;
}
if (N < maxThreadsPerBlock){
   threadsPerBlock = N;
   numBlocks = (N + N - 1) / N;
}
else{
   threadsPerBlock = maxThreadsPerBlock;
   numBlocks = (N + maxThreadsPerBlock - 1) / maxThreadsPerBlock;
}

//***************************************
// Transient solution process
//***************************************
std::clock_t start;
for (int itime = 0; itime < timeSimu1; itime++){
   // Start of the time controller
   if (multiSys == 1){
      start = std::clock();
   }
   //*********************************
   // Kernel call
   //*********************************
   //*****************************************************************************************
   if (multiSys == 1){
      if ((upper_limit_on == 1) && (lower_limit_on == 1)){
         par_var_TxTy << < numBlocks, threadsPerBlock >> > (storage, def_med, d_Loc, cols, rows,
         d_initial_condition, d_upper_limit, d_lower_limit, d_DFx, d_DFy, d_Sy, d_CT, d_Tx, d_Ty, d_lambda, N);
      }
   }
   else{
      if ((upper_limit_on == 1) && (lower_limit_on == 1)){
         par_var_TxTy << < numBlocks, threadsPerBlock >> > (storage, def_med, d_Loc, cols, rows,
         d_initial_condition, d_upper_limit, d_lower_limit, d_DFx, d_DFy, d_Sy, d_CT, d_Tx, d_Ty, d_lambda, N);
      }
      else{
         par_const_TxTy << <numBlocks, threadsPerBlock >> >(storage, def_med, d_Loc, d_DFx, d_DFy,
         diff_upper_lower, d_Sy, d_CT, d_Tx, d_Ty, d_lambda, N);
      }
   }
   if ((multiSys == 1) && (save_velocity == 1)){
   //**************************************
      gw_velocity << <numBlocks, threadsPerBlock >> >(def_med, d_CT, cols, rows, d_Loc, d_initial_condition,
      d_dy, d_dx, d_DFx, d_DFy, d_ne, d_Vx_vec, d_Vy_vec, N);
      cudaMemcpy(h_Vx_vec, d_Vx_vec, Nval*sizeof(float), cudaMemcpyDeviceToHost);
      cudaMemcpy(h_Vy_vec, d_Vy_vec, Nval*sizeof(float), cudaMemcpyDeviceToHost);
   //**************************************
   }
   // Linear system construction kernel (organization of data in vector format)
   ClassMatPOS << <numBlocks, threadsPerBlock >> >(def_med, velocity_vec, diffusion, advection, d_lambda, decay,
   mob, d_CT, d_Loc, d_IDrow, d_IDcolumn, d_FD_1, d_FD_2, cols, rows, N, Nval, d_Tx, d_Ty, d_Vx, d_Vy, d_dx, d_dy,
   dt);
   //*******************************************
   //*******************************************
   // Copying the GPU results to the CPU
   //*******************************************
   cudaMemcpy(h_FD_1, d_FD_1, dimSys*sizeof(float), cudaMemcpyDeviceToHost);
   cudaMemcpy(h_FD_2, d_FD_2, dimSys*sizeof(float), cudaMemcpyDeviceToHost);
   cudaMemcpy(h_IDrow, d_IDrow, dimSys*sizeof(int), cudaMemcpyDeviceToHost);
   cudaMemcpy(h_IDcolumn, d_IDcolumn, dimSys*sizeof(int), cudaMemcpyDeviceToHost);
   int non_null = 0;
   for (int i = 0; i < dimSys; i++){ if (h_FD_1[i] != 0){ non_null++; } }
   cusp::coo_matrix<int, ValueType, MemorySpaceh> A(Nval, Nval, non_null);
   cusp::coo_matrix<int, ValueType, MemorySpaceh> B(Nval, Nval, non_null);
   cont = -1;
   for (int i = 0; i < dimSys; i++){
      if (h_FD_1[i] != 0){
         cont = cont + 1;
```

```
            // Matrix A
            A.row_indices[cont] = h_IDrow[i];
            A.column_indices[cont] = h_IDcolumn[i];
            A.values[cont] = h_FD_1[i];
            // Matrix B
            B.row_indices[cont] = h_IDrow[i];
            B.column_indices[cont] = h_IDcolumn[i];
            B.values[cont] = h_FD_2[i];
        }
    }
    // Sort by row indices
    A.sort_by_row();
    B.sort_by_row();
    // Conversion to hybride format (hyb) and copy to GPU
    cusp::hyb_matrix<int, ValueType, MemorySpace> d_A(A);
    cusp::hyb_matrix<int, ValueType, MemorySpace> dA(B);
    // Starting the Time Controller
    if (multiSys == 0){
        start = std::clock();
    }
    for (int itime1 = 0; itime1 < timeSimu2; itime1++){
        // Starting the Time Controller
        //if (multiSys == 0){
        // start = std::clock();
        //}
        if ((multiSys == 0) && (save_velocity == 1)){
        //****************************************
            // Kernel call for speed calculation
            gw_velocity << <numBlocks, threadsPerBlock >> >(def_med, d_CT, cols, rows, d_Loc, d_initial_condition,
            d_dy, d_dx, d_DFx, d_DFy, d_ne, d_Vx_vec, d_Vy_vec, N);
            cudaMemcpy(h_Vx_vec, d_Vx_vec, Nval*sizeof(float), cudaMemcpyDeviceToHost);
            cudaMemcpy(h_Vy_vec, d_Vy_vec, Nval*sizeof(float), cudaMemcpyDeviceToHost);
        //****************************************
        }
        //Groundwater source term - base flow
        if (base_flow == 1){
            subtract_vet << < numBlocks, threadsPerBlock >> >(d_upper_limit, d_initial_condition, N, d_UpperVar);
            gw_source_baseflow << <numBlocks, threadsPerBlock >> >(def_med, d_Loc, d_UpperVar, d_dx, d_dy, d_Sy,
            d_CT, d_CdrV, d_W, d_Qbase, Cdr, N, dt);
            cudaMemcpy(h_W, d_W, Nval*sizeof(float), cudaMemcpyDeviceToHost);
            cudaMemcpy(h_Qbase, d_Qbase, Nval*sizeof(float), cudaMemcpyDeviceToHost);
            for (int i = 0; i < Nval; i++){
                Wn[i] = h_W[i];
            }
        }
        // Logistic growth source term
        // *****************************************************
        if (logistic_growth == 1){
            Wn = logistic_term(Mrub, ru, un, diff_unb, Wn, H);
        }
        // Multiplicação b = dB*H
        cusp::multiply(dA, H, b);               //dB
        // Somas Ws = b+Wn;
        cusp::blas::axpby(b, Wn, Ws, 1, 1);

        // set stopping criteria:
        //  iteration_limit    = 100
        //  relative_tolerance = 1e-20
        //  absolute_tolerance = 0
        //  verbose            = true
        cusp::convergence_monitor<ValueType>monitor(b, 1000, 1e-20, 0);
        // set preconditioner (identity)
        //cusp::identity_operator<float, cusp::device_memory> M(Nval, Nval);
        // Solução do sistema linear
        cusp::krylov::bicgstab(d_A, H, Ws, monitor);// , M); // Stabilized Biconjugado Gradient    //dA
        //float sum = 0; //total sum
        int ks = -1;
        for (int j = 0; j < N; j++){
            if (h_CT[j] != 0) {
                ks++;
                h_initial_condition[j] = H[ks];
                //sum += H[ks];
                if (h_CT[j] < 0){
                    H[ks] = h_Hconst[ks];
                }
            }
        }
        cudaMemcpy(d_initial_condition, h_initial_condition, N*sizeof(float), cudaMemcpyHostToDevice);
        //Time
        duration = (std::clock() - start) / (double)CLOCKS_PER_SEC;
        std::cout << "printf: " << duration << '\n';
```

```
if ((save_times == 1) && (save_final_result == 0) && (save_result_sequence == 0)){
   oud << duration;
   ittime = oud.str();
   if (multiSys == 1){
      out << itime;
   }
   else {
      out << itime1;
   }
   tempo = out.str();
   out.str("");
   oud.str("");
   // Save Times
   FILE *TimeOutput;
   dirTimeOutput = "db\\" + dir_parameters + "\\output\\TimeSystemSolution\\TimeSimu_" + tempo + ".txt";
   TimeOutput = fopen(dirTimeOutput.c_str(), "w");
   fprintf(TimeOutput, " %f\n", duration);
   fclose(TimeOutput);
}
if ((save_final_result == 1) || (save_result_sequence == 1)){
   if (multiSys == 1){
      if (save_result_sequence == 1){
         off_save_sequence = itime % save_step;
      }
   }
   else {
      if (save_result_sequence == 1){
         off_save_sequence = itime1 % save_step;
      }
   }
   if (save_final_result == 1){
      off_save_sequence = 1;
   }
   if ((off_save_sequence == 0) || (itime == timeSimu - 1) || (itime1 == timeSimu - 1)){
      oud << duration;
      ittime = oud.str();
      if (multiSys == 1){
         out << itime;
      }
      else {
         out << itime1;
      }
      tempo = out.str();
      out.str("");
      oud.str("");
      if (save_times == 1){
      // Save Times
         FILE *TimeOutput;
         dirTimeOutput = "db\\" + dir_parameters + "\\output\\TimeSystemSolution\\TimeSimu_" + tempo +
         ".txt";
         TimeOutput = fopen(dirTimeOutput.c_str(), "w");
         fprintf(TimeOutput, " %f\n", duration);
         fclose(TimeOutput);
      }
      FILE *writeU;
      FILE *writeVxtxt;
      FILE *writeVytxt;
      std:string outU, outVxtxt, outVytxt, NameLayers;
      if ((file_txt == 1) && (save_velocity == 1)){
         outVxtxt = "db\\" + dir_parameters + "\\output\\velocity_x_txt\\Vx_" + tempo + ".txt";
         writeVxtxt = fopen(outVxtxt.c_str(), "w");
         outVytxt = "db\\" + dir_parameters + "\\output\\velocity_y_txt\\Vy_" + tempo + ".txt";
         writeVytxt = fopen(outVytxt.c_str(), "w");
         fprintf(writeVxtxt, "north: %s\n", &north);
         fprintf(writeVxtxt, "south: %s\n", &south);
         fprintf(writeVxtxt, "east: %s\n", &east);
         fprintf(writeVxtxt, "west: %s\n", &west);
         fprintf(writeVxtxt, "rows: %d\n", rows);
         fprintf(writeVxtxt, "cols: %d\n", cols);
         fprintf(writeVytxt, "north: %s\n", &north);
         fprintf(writeVytxt, "south: %s\n", &south);
         fprintf(writeVytxt, "east: %s\n", &east);
         fprintf(writeVytxt, "west: %s\n", &west);
         fprintf(writeVytxt, "rows: %d\n", rows);
         fprintf(writeVytxt, "cols: %d\n", cols);
      }
      if ((file_txt == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
         outU = "db\\" + dir_parameters + "\\output\\solution_txt\\U_solution_" + tempo + ".txt";
         NameLayers = "U_solution_" + tempo + ".txt";
         writeU = fopen(outU.c_str(), "w");
         fprintf(writeU, "north: %s\n", &north);
         fprintf(writeU, "south: %s\n", &south);
         fprintf(writeU, "east: %s\n", &east);
         fprintf(writeU, "west: %s\n", &west);
         fprintf(writeU, "rows: %d\n", rows);
         fprintf(writeU, "cols: %d\n", cols);
      }
      if ((file_mtx == 1) || (file_txt == 1)){
         int km = -1;
         int km1 = -1;
         for (int jm = 0; jm <= rows - 1; jm++) {
            for (int im = 0; im <= cols - 1; im++) {
               km = km + 1;
               if (h_CT[km] != 0){
                  km1 = km1 + 1;
                  if ((file_mtx == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
                     // Matriz Umtx
                     Umtx.row_indices[km1] = jm;
                     Umtx.column_indices[km1] = im;
                     Umtx.values[km1] = H[km1];
                  }
                  if ((file_mtx == 1) && (save_velocity == 1)){
                     Vx_vec(jm, im) = h_Vx_vec[km1];
                     Vy_vec(jm, im) = h_Vy_vec[km1];
                  }
                  if ((file_txt == 1) && (save_velocity == 1)){
                     if (im == (cols - 1)){
                        fprintf(writeVxtxt, " %f\n", h_Vx_vec[km1]);
                        fprintf(writeVytxt, " %f\n", h_Vy_vec[km1]);
                     }
                     else{
                        fprintf(writeVxtxt, " %f", h_Vx_vec[km1]);
                        fprintf(writeVytxt, " %f", h_Vy_vec[km1]);
                     }
                  }
                  if ((file_txt == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
                     U2d[jm][im] = H[km1];
                     if (im == (cols - 1)){
                        fprintf(writeU, " %f\n", U2d[jm][im]);
                     }
                     else{
                        fprintf(writeU, " %f", U2d[jm][im]);
                     }
                  }
               }
               else{
                  if ((file_mtx == 1) && (save_velocity == 1)){
                     // Value that will occupy the cells that do not belong to the computational domain
                     Vx_vec(jm, im) = 0;
                     Vy_vec(jm, im) = 0;
                  }
                  if ((file_txt == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
                     //Value that will occupy the cells that do not belong to the computational domain
                     U2d[jm][im] = 0;
                     if (im == (cols - 1)){
                        fprintf(writeU, " %f\n", U2d[jm][im]);
                     }
                     else{
                        fprintf(writeU, " %f", U2d[jm][im]);
                     }
                  }
                  //Value that will occupy the cells that do not belong to the computational domain
                  float bf = 0;
                  if ((file_txt == 1) && (save_velocity == 1)){
                     if (im == (cols - 1)){
                        fprintf(writeVxtxt, " %f\n", bf);
                        fprintf(writeVytxt, " %f\n", bf);
                     }
                     else{
                        fprintf(writeVxtxt, " %f", bf);
                        fprintf(writeVytxt, " %f", bf);
                     }
                  }
               }
            }
         }
      }
      if ((file_txt == 1) && (save_velocity == 1)){
         fclose(writeVxtxt);
         fclose(writeVytxt);
      }
      if ((file_txt == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
```

```
                     fclose(writeU);
                     fprintf(writeNameLayers, "%s\n", NameLayers.c_str());
                  }
                  if ((file_mtx == 1) && ((save_final_result == 1) || (save_result_sequence == 1))){
                     // Generating Outputs Resp.mtx
                     cusp::io::write_matrix_market_file(Umtx, "db\\" + dir_parameters +
                     "\\output\\solution_mtx\\Resp_H_" + tempo + ".mtx");
                     if (save_velocity == 1) {
                        cusp::io::write_matrix_market_file(Vx_vec, "db\\" + dir_parameters +
                        "\\output\\velocity_x_mtx\\Vx_" + tempo + ".mtx");
                        cusp::io::write_matrix_market_file(Vy_vec, "db\\" + dir_parameters +
                        "\\output\\velocity_y_mtx\\Vy_" + tempo + ".mtx");
                     }
                  }
               }
            }
         }
      }
      if ((file_txt == 1) && ((save_final_result == 1)||(save_result_sequence==1))){
         fclose(writeNameLayers);
      }
      // *************************
      // Clear memory
      // *************************
      cudaFree(d_initial_condition);    free(h_initial_condition);
      if ((upper_limit_on == 1) && (lower_limit_on == 1)){
         cudaFree(d_upper_limit);            free(h_upper_limit);
         cudaFree(d_lower_limit);            free(h_lower_limit);
         cudaFree(d_UpperVar);               free(h_UpperVar);
         cudaFree(d_LowerVar);               free(h_LowerVar);
      }
      cudaFree(d_Sy);                   free(h_Sy);
      cudaFree(d_DFx);                  free(h_DFx);
      cudaFree(d_DFy);                  free(h_DFy);
      cudaFree(d_Vx);                   free(h_Vx);
      cudaFree(d_Vy);                   free(h_Vy);
      cudaFree(d_Vx_vec);               free(h_Vx_vec);
      cudaFree(d_Vy_vec);               free(h_Vy_vec);
      cudaFree(d_Tx);
      cudaFree(d_Ty);
      cudaFree(d_dx);                   free(h_dx);
      cudaFree(d_dy);                   free(h_dy);
      //cudaFree(d_x);                  //free(h_x);
      //cudaFree(d_y);                  //free(h_y);
      cudaFree(d_Hconst);               free(h_Hconst);
      cudaFree(d_W);                    //free(h_W);
      cudaFree(d_CdrV);                 //
      cudaFree(d_Qbase);                //free(h_Qbase);
      cudaFree(d_CT);                   free(h_CT);
      //cudaFree(d_Loc);                  free(h_Loc);
      cudaFree(d_FD_1);                 free(h_FD_1);
      cudaFree(d_FD_2);                 free(h_FD_2);
      cudaFree(d_IDrow);                free(h_IDrow);
      cudaFree(d_IDcolumn);             free(h_IDcolumn);
   }
   return 0;
}
```